UNIVERSITY OF HERTFORDSHIRE

Faculty of Engineering & Information Science

BATCHELOR OF ENGINEERING DEGREE WITH HONOURS IN ELECTRICAL AND ELECTRONIC ENGINEERING

Project Report

Dual processor, shared memory, 68000 system

Darren Lee Witts

April 2003

University of Hertfordshire

1. Abstract

This report is a detailed description of the work carried out in order to develop a prototype of a shared memory module that is compatible with the MC68000 microprocessor training system by Flight Electronics Ltd.

The module is intended to be used together with two Flight MC68000 MKII training systems as an educational aid for computer architecture courses where the syllabus contains parallel processing.

An account is given of all aspects of the project including design, manufacture of a printed circuit board, testing and debugging together with research and associated project management issues.

The report is written for anybody who wishes to gain an understanding of the work that has taken place during the project and for those wishing to carry out further development.

University of Hertfordshire

2. Acknowledgements

There are many people who contributed to the success of this project by providing help and guidance, without any of which the project would not have reached the stage it has. The help of all such people is gratefully acknowledged.

In particular, the help of the following people at the University of Hertfordshire deserve my gratitude.

Professor Reza Sotudeh (Project Supervisor) for providing motivation, ideas and continued support throughout the project.

Mr Tony Crook (Technical Manager) for assisting with many practical decisions and guidance for designing the printed circuit board and subsequent build.

Mr Stephen Passmore (Technician) for providing assistance with access to the microprocessor laboratory and making available the necessary equipment when needed.

University of Hertfordshire

3. Contents

1. Abstract 2
2. Acknowledgements 3
3. Contents 4
3.1 Table of figures 5
4. Glossary and Abbreviations 6
5. Introduction 7
5.1 Design brief and discussion 7
5.2 An educational tool 7
5.3 Project aims and objectives 8
5.4 This report 8
6. Background 9
6.1 The Motorola 68000 9
6.2 Flight 68000 MK-II training system 10
6.3 Flynn's Taxonomy 11
6.4 Computer memory 13
6.5 Shared memory module overview 14
6.5.1 Arbitration 14
6.5.2 Buses 16
7. Research 17
7.1 Literature search 17
7.2 Hardware research 17
7.3 Software research 17
8. Circuit Design 18
8.1 Address decoding 18
8.2 Simulation of address decoding design 21
8.3 Output latch 21
8.4 Arbitration 23
8.5 Bus signal isolation 23
8.6 DTACK and RAM 24
8.7 Power supply 24
9. Printed Circuit Board and Mechanics 25
10. Power Test Results and Debugging 27
10.1 Methodology for power tests 27
10.2 Results and rectifications for power tests 27
10.2.1 Visual inspection 27
10.2.2 Voltage and current measurements 27
10.2.3 Voltage at IC pins 27
10.2.4 Voltage on bus connector 27
11. Functional Test Results and Debugging 28
11.1 Methodology for functional tests 28
11.2 Results and rectifications for functional tests 28
11.2.1 Memory read and write 28
11.2.2 Arbitration test 34
11.2.3 Address bus walking 1s test 34
11.2.4 Data bus walking 1s test 35
11.2.5 SISD test 35
11.2.6 SIMD test 35
11.2.7 MISD test 37
11.2.8 MIMD test 37
11.2.9 C Compiler 37
12. Software 39
13. Conclusions 40

University of Hertfordshire

14. Further development 41
14.1 PCB improvement 41
14.2 Circuit improvement 41
14.3 Software 41
15. Project management 42
15.1 Resources 42
15.2 Cost analysis 42
15.3 Time analysis 42
15.4 Initial Gantt chart 44
15.5 Final Gantt chart 45
16. References 46
17. Bibliography 47
18. List of Appendices 49

3.1 Table of figures

Figure 1. 68000 function codes 9
Figure 2. The Motorola 68000[1] 10
Figure 3. Flight 68000 connection to personal computer 11
Figure 4. F68k PI/T addresses 11
Figure 5. SISD Architecture 12
Figure 6. MIMD Architecture 12
Figure 7. SIMD Architecture 12
Figure 8. MISD Architecture 13
Figure 9. Expanded/ Extended memory 14
Figure 10. Truth table of first come first served arbiter 15
Figure 11. Block diagram of arbitration 15
Figure 12. Shared memory module block diagram 16
Figure 13. Address decoding bits 18
Figure 14. Logic to force A_{16} low when in extended mode 19
Figure 15. Address decoder circuit 19
Figure 16. RAM chip select logic truth table 20
Figure 17. RAM chip select circuit diagram 20
Figure 18. RAM chip select simulated waveform 21
Figure 19. Output latch address decoding 22
Figure 20. Output latch circuit 22
Figure 21. PCB width and correct orientation 25
Figure 22. Initial C test program 28
Figure 23. Monitor terminal window showing read/write command 29
Figure 24. Monitor terminal window following address bus corrections 30
Figure 25. Modified address decoding 31
Figure 26. SMM under test showing modification 32
Figure 27. Address decoding waveform 32
Figure 28. Bus arbitration waveform 34
Figure 29. Table to show values used for walking 1s test 35
Figure 30. SIMD test program 36
Figure 31. SIMD test program subroutine 36
Figure 32. Disassembled test program 37
Figure 33. Compiler error message 38
Figure 34. Software hierarchy 39

University of Hertfordshire

4. Glossary and Abbreviations

Abbreviation	Full meaning	Description
~	Inversion	
A	Amperes	
AS	Address Strobe	68000 output signal
A_x	Address line *x*	Address bus signal
CE	Chip Enable	Logic chip input
CPU	Central Processing Unit	
CS	Chip Select	Logic chip input
DC	Direct Current	
DTACK	Data Transfer Acknowledge	68000 control bus signal
D_x	Data line x	Data bus signal
EDS	Embedded Development Studio	Flight 68000 C compiler
EPROM	Erasable Read Only Memory	
F68k	Flight 68000 MK-II training system	
FCx	Function Code x	68000 status output signal
FIFO	First In First Out	
I/O	Input/Output	
IC	Integrated Circuit	
LAN	Local Area Network	
LDS	Lower Data Strobe	68000 control bus signal
LED	Light Emitting Diode	
MIMD	Multiple Instructions, Multiple Data	Flynn's Taxonomy classification
MISD	Multiple Instructions, Single Data	Flynn's Taxonomy classification
MO	Memory Open	Flight monitor command
PC	Personal Computer	
PCB	Printed Circuit Board	
PI/T	Peripheral Interface/Timer	Input/Output port controller
R/W	Read/Write	68000 input signal
RAM	Random Access Memory	
ROM	Read Only Memory	
SIMD	Single Instruction, Multiple Data	Flynn's Taxonomy classification
SISD	Single Instruction, Single Data	Flynn's Taxonomy classification
SPICE	Simulation Program with Integrated Circuit Emphasis	Method of simulating a circuits behaviour on a computer
UDS	Upper Data Strobe	68000 control bus signal
V	Volts	

Unive

5. Introduction

5.1 Design brief and discussion

> *Design a system to enable two Flight MC68000 – MKII training systems to operate as a dual processor system with the tightest form of coupling (shared memory). The design needs to interface to the existing training boards and be used in conjunction with the training boards for educational/training purposes.*

Dual processor computer systems fall into the categories or multiprocessing and parallel architectures. The parallel architecture allows a function to be divided into smaller tasks that can be carried out concurrently thereby producing results faster than single processor systems.

In order for tasks to be completed by different processors those processors must have a means to communicate with each other to return results or pass data and instructions. Common methods of communication include serial and parallel ports or LANs such as Ethernet. These methods are termed *loosely coupled* due to the limited data transfer speeds that can be obtained for the distances over which they operate. First in first out (FIFO) shift registers are more *tightly coupled* because of their small propagation delay however the output from the shift register must normally be transferred into memory before being read by a processor. Shared memory is considered the tightest form of coupling since the only propagation delays involved in a transaction are the read and write times of the memory being used and any interface circuitry required.

5.2 An educational tool

The use of small microprocessor based training systems are widely used in educational laboratories to teach students the concepts of computer architecture and programming of systems in both high and low level languages. Not so common is the use of such systems for teaching the concepts of parallel processing.

Loosely coupled parallel processing can easily be demonstrated by interfacing standard personal computers at the serial, parallel or network ports. On the other hand, tightly coupled parallel processing can only be achieved with specifically designed multiprocessor systems.

The University of Hertfordshire's microprocessor training laboratory contains Motorola MC68000 microprocessor training systems by Flight Electronics Ltd. Flight's catalogue does not include a multiprocessor training system and research conducted at the beginning of this project did not reveal any such systems by other manufacturers. The absence of multiprocessor systems for educational use may be attributed to the significant cost increase of a system containing two or more processors. Without a training system the student is at a disadvantage since the practical aspects cannot be explored or the theory related and tried.

The price of memory is no longer a major limiting factor of a systems design. Acceptable memory prices together with the fact that sharing memory between two processors provides the tightest form of coupling lead to the designing of a shared memory module intended for use within the microprocessor laboratory at the University of Hertfordshire. The Flight Electronics 68000 training systems used within the laboratory contain sufficient expansion ports to access all bus signals (see Appendix M page 5) and therefore accommodate a shared memory module. The Motorola MC68000 microprocessor also has the ability to initiate memory transactions asynchronously which adds to the feasibility of a shared memory module.

University of Hertfordshire

A shared memory module counteracts the cost implications associated with two microprocessors by interfacing between two separate training systems. For the study of single processor architectures the boards act as the original stand-alone products but for studying dual processing the separate boards can be plugged into the shared memory module and a full parallel architecture is then at the students disposal.

5.3 Project aims and objectives

Listed below are the key objectives that need to be considered and accomplished in order to achieve the aim of designing a working prototype of a shared memory module which satisfies the design brief.

- Research similar or related projects and literature
- Learn about and research parallel forms of computer architecture
- Learn about and research how two processors can work in parallel with the tightest form of coupling (shared memory)
- Learn about and research the MC68000 microprocessor
- Learn about and research the Flight 68000 training systems available within the university, in particular how to use and interface to them
- Design and build circuitry to interface to the MC68000 training system
- Design and program software suitable for operating the completed hardware
- Test completed build of the system

5.4 This report

This report has been written primarily for two types of people. Firstly so that any person with a basic understanding of electronics can understand the work that has taken place and the results that have been reached and secondly so that an electronics engineer can duplicate the work or gain a thorough understanding in order to carry out further research or development. Recommendations for further development are given in chapter 14.

The report is structured so that the reader can follow through the chapters and build up an understanding of the problems faced and how they were solved and integrated into the design. To do this, the first chapters give a background into related theory. Alternatively, the informed reader can skip directly to the detailed design chapters to gain the same understanding.

Reference will be made throughout the text to signals being ‘asserted’. If a signal is active low then when asserted the signal should be considered at a logic low level. If a signal is active high then when asserted the signal should be considered at a logic high level.

Every effort has been made within the text to avoid any confusion arising between references made to the shared memory module board and the Flight 68000 microprocessor board and also between shared memory and local memory.

University of Hertfordshire

6. Background

6.1 The Motorola 68000

The 68000 microprocessor is manufactured by Motorola Incorporated. It contains a 16-bit wide data bus and an internal address bus of 24-bits. The 24 bit address bus means that the 68000 can address up to 2^{24}=*16Mbytes* or *8Mwords* of RAM, ROM or memory mapped peripherals.

Externally the address bus does not include the least significant bit, A_0, which means that odd addresses are not addressable. Instead of A_0 two data strobes are provided, UDS (Upper Data Strobe) and LDS (lower Data Strobe). These data strobes are active low signals and indicate whether the upper or lower byte of a data word is being addressed. If an entire word is being addressed then both data strobes are asserted.

UDS and LDS can be used within a systems address decoding. At the simplest level UDS and LDS can be connected directly to the chip select (or chip enable) input of two 8-bit wide (data bus) memory devices, thus the upper and lower bytes of the word are stored in physically separate devices.

An important feature of the 68000 that differentiates it from other microprocessors such as the Philips 80C51 or 80C552 is its' ability to carry out asynchronous bus cycles. A handshaking signal is used as an input to the 68000 to inform it when an external memory device or peripheral has provided or responded to data on the bus. This handshaking signal is called Data Transfer Acknowledge (DTACK) and is active low. If DTACK is not asserted when the 68000 expects it then wait states are introduced to the bus cycle until it is asserted. With a synchronous bus, memory devices would have to be fast enough to have data ready when the processor expects it and peripherals fast enough to act on data before the processor removes it from the bus. The asynchronous bus therefore allows the 68000 to be used with slower memory devices and peripherals than would otherwise be possible.

There are three signals called function codes (FC0, FC1 and FC2) that are output from the 68000. The functions codes provide a means of determining what state the 68000 is currently in as shown in Figure 1. The function codes and the current address on the bus are considered valid when the address strobe (AS) is asserted active low by the processor.

Function Code			
FC0	**FC1**	**FC2**	**Meaning**
0	0	0	Reserved
0	0	1	User data being accessed
0	1	0	User program being accessed
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor data being accessed
1	1	0	Supervisor program being accessed
1	1	1	Interrupt being processed

Figure 1. 68000 function codes

One further signal of importance to bus transactions is the read/write output ($R/\overline{W}$). When this output goes low it indicates that the processor has started a write cycle. When the output is high is means that a read cycle is in progress. Figure 2 shows the 68000 outline and pin details.

University of Hertfordshire

Figure 2. The Motorola 68000[1]

6.2 Flight 68000 MK-II training system

The Flight 68000 MK-II training system (F68k) is a product manufactured by Flight Electronics Ltd and is intended to be used within an educational or training environment for students to gain an introduction to microprocessors, and in particular the 68000.

The F68k operates a 10*MHz* clock speed and contains 16*kbytes* of local memory split into upper and lower byte physical devices which are expandable to 512*kbytes*. The F68k microprocessor board contains an input/output port which can be interfaced to application boards containing LEDs, motors, analogue to digital converters, etc.

The F68k also caters for the more advanced student by providing a bus expansion port which connects to all of the 68000s 64 pins with the exception of V_{CC}. This enables external peripherals and extra memory to be installed on user designs which interface to the bus expansion port.

The standard monitor provided with the F68k is located on upper and lower byte EPROMs. The monitor provides a command line interface which is accessible through any terminal window reading the F68k serial port. The most common configuration is by connecting the serial port of a personal computer to the F68k.

The monitor provides an assembler (AS) and a disassembler (DS) and other commands[2] such as memory open (MO), block fill (FI) and block print (PR). The assembler enables assembly language programs to be entered line by line. MO, FI and PR can be used to examine and alter the contents of specific memory locations, which is useful for testing if a read or a write occurs successfully.

There is also a C compiler available for the F68k. The compiler replaces the monitor EPROMs on the F68k microprocessor board and the terminal window on the personal computer is

University of Hertfordshire

replaced with a program called Embedded Development Studio (EDS) by Crossware Associates.

Figure 3 shows the setup between the F68k and a personal computer running a serial port terminal or the EDS compiler.

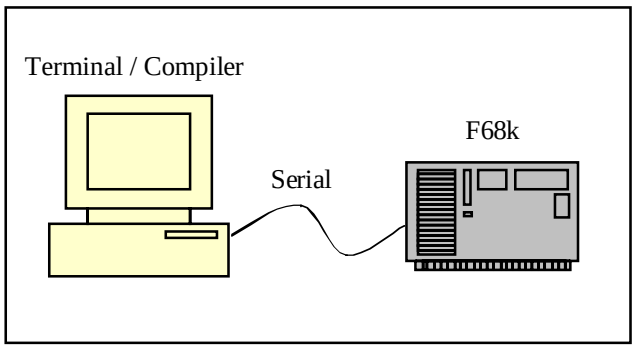


Figure 3. Flight 68000 connection to personal computer

Also included on board the F68k is a 68230 peripheral interface/timer (PI/T). The PI/T provides the user of the F68k with I/O lines and timer functions for interfacing to the outside world. The PI/T port is compatible the F68k application board and other application boards designed by users. There are two ports on the PI/T used as inputs and outputs which are mapped into the memory map starting at 800001[3]. The basic addresses of these ports are given in Figure 4.

PI/T	Port A address	Port B address
Data Direction Register	800005	800007
Control Register	80000D	80000F
Data Register	800011	800013

Figure 4. F68k PI/T addresses

6.3 Flynn's Taxonomy

One of the most common classifications of parallel architectures was stated by Michael J Flynn in 1966 and is known as Flynn's Taxonomy[4]. This taxonomy considers the two bit streams within computer architectures; instructions (opcodes) and data (operands). The taxonomy details four classifications that define where in the architecture the instructions and the data reside, irrespective of whether they are loose or tightly coupled.

Single Instruction, Single Data (SISD) is the classification defining any single processor system such as a standard personal computer in which data and instructions are stored in the same memory block, as shown in Figure 5.

University of Hertfordshire

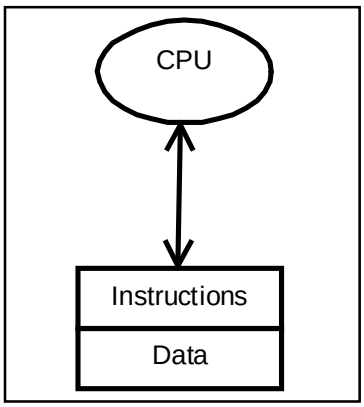


Figure 5. SISD Architecture

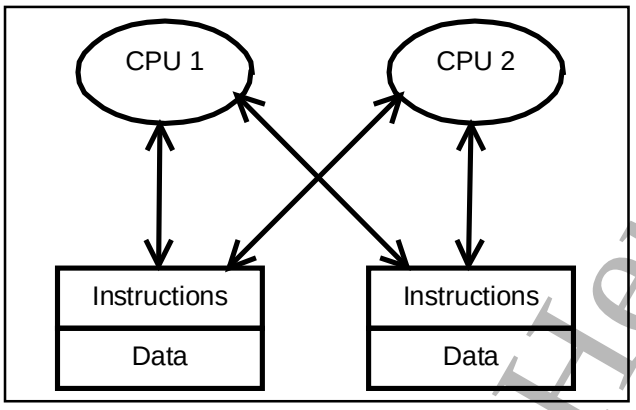


Figure 6. MIMD Architecture

Multiple Instructions, Multiple Data (MIMD) is a very broad classification and includes any collection of SISD systems that can communicate with each other, for example a number of PCs on a local area network. MIMD is represented in Figure 6.

Single Instruction, Multiple Data (SIMD) specifies systems in which the instructions are stored in a shared location and can be accessed by multiple processors, each executing those *same instructions* on their own *individual data*. SIMD is represented in Figure 7.

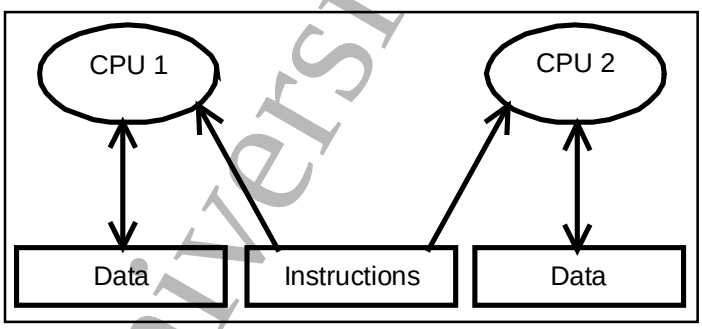


Figure 7. SIMD Architecture

Multiple Instructions, Single Data (MISD) is the opposite of SIMD in that it specifies systems in which the data is stored in a shared location and can be accessed by multiple processors, each executing their own *individual instructions* on that *same data*. MISD is represented in Figure 8.

University of Hertfordshire

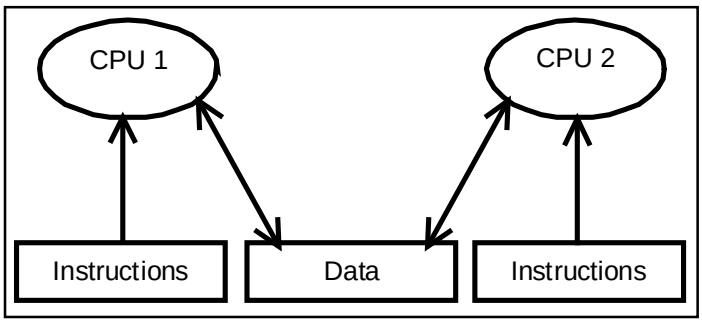


Figure 8. MISD Architecture

Coupling two processors with shared memory enables those processors to communicate via that shared memory, storing and retrieving data and instructions as and where necessary. Therefore it is perfectly possible to achieve each of the above classifications with the shared memory module that is the focus of this project.

Any system employing multiple processors with shared memory must also include local memory that is dedicated to each processor. Without any local memory each processor might have no choice but to wait before gaining access to the shared memory, thereby defeating the object of faster parallel processing.

6.4 Computer memory

There are other aspects of computer architecture associated with memory that are not specific to parallel architectures. One such aspect is that of paging, or expanded memory.

Expanded memory is more common in older systems where the number of address bits on a microprocessor were limited however it does still have uses, particularly for educational purposes where it is sometimes necessary to show the operation of older systems.

Expanded memory allows more than one block of RAM (pages) to be located at a particular address range. Switching is then performed in order to select the page required at any given time.

Paging also has some applications in data security because the main system RAM can be copied to another page as a backup. A process could be written to periodically perform this backup; this is often called ghosting or shadowing. Similar backup systems exist mainly for fixed storage, such as computer hard disks in servers.

For applications where extended memory is not required it is possible to relocate the additional pages within the first page memory map. This is termed extended memory and it is possible to provide a method of enabling the user to configure the memory as expanded or extended as dictated by their particular application.

Figure 9 shows the location of the second page of memory in a two page system depending upon whether extended or expanded mode is being used.

University of Hertfordshire

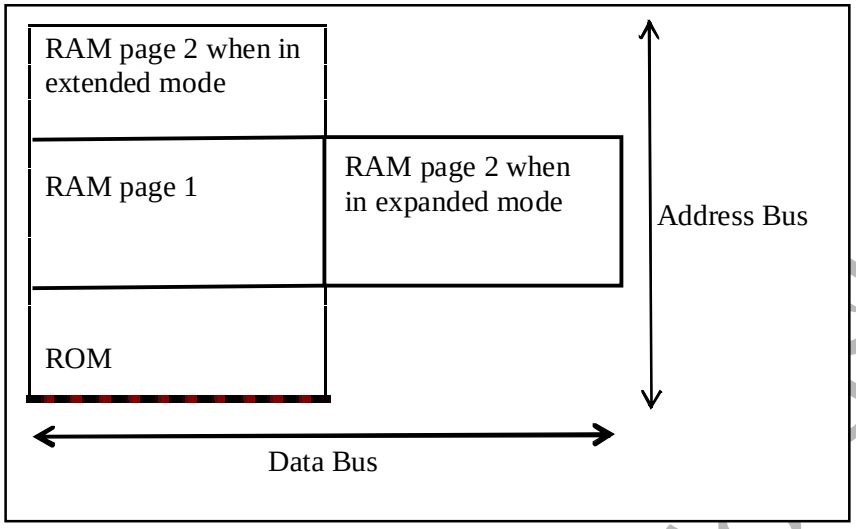


Figure 9. Expanded/ Extended memory

6.5 Shared memory module overview

6.5.1 Arbitration

As the system contains dual processors and both of those processors require access to a shared media (memory), it is clear that there will be contention for that media when both processors require access simultaneously. To avoid contention and subsequent corruption of data occurring, a form of arbitration is needed to arbitrate the request signals from each processor.

When a processor generates a request for the shared memory it is the arbiters function to grant access if certain rules are satisfied. Arbitration rules include the following:

- First come first served
- Fixed priority
- Variable priority
- Time scheduled (round robin)
- Length of time access is needed for

First come, first served grants access to the first processor that makes a request and holds the other processor back until the first has completed its use of the shared memory.

A prioritised arbiter grants access to the most highly prioritised processor. The processor priorities may be fixed or they may be dynamic, changing depending upon the task that is currently being executed.

A time scheduled arbitration scheme grants access to each processor for a fixed period of time which may or may not be enough time for the processor to complete the tasks it is executing.

Arbitrating on the length of time that access is needed for really means assigning a priority depending upon the length of time that a processor states it requires access for when it makes a request. Normally the processor that asks for the shortest length of time is granted access so causing fewer contentions.

All of the arbitration schemes given above, with the exception of first come first served, are capable of being implemented in software by the two processors involved communicating their priorities or time requirements to each other. However, for two processors to exchange such

University of Hertfordshire

information they must access the shared memory and leave that information in memory for the other processor to read. In accessing the memory contention may occur and therefore a method of hardware arbitration such as first come first served is essential.

With a first come first served arbitrator one processor can inform the other via software that it will be using shared memory for a specific amount of time and therefore not to attempt access. If the other processor does still attempt access, maybe due to a poorly written program or a corruption in receiving a message, then the hardware arbiter will deny it that access and no corruption of data will occur. The advantage of using such a hardware and software arbitration scheme would be that the second processor will not waste time waiting for access.

Figure 10 shows the truth table for an active low, first come fist served arbiter. The block diagram in Figure 11 shows the basic interconnections and logical structure of the arbitration.

A request	B request	A grant	B grant	Description
0	0	First come first served		Both request access simultaneously or when one has already been granted access
0	1	0	1	A requests access, no contention
1	0	1	0	B requests access, no contention
1	1	1	1	Neither request access, contention

Figure 10. Truth table of first come first served arbiter

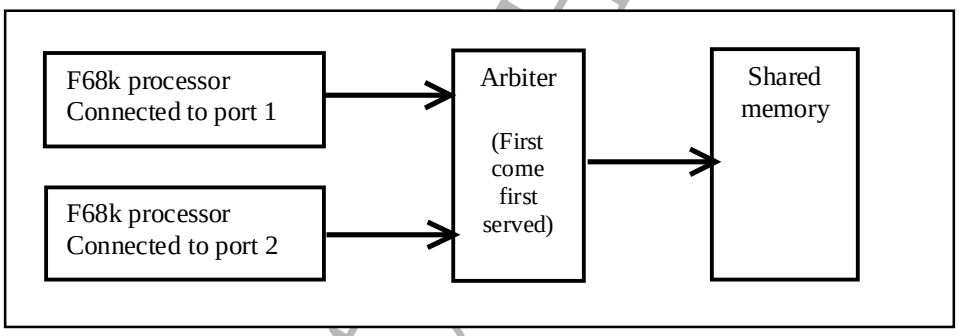


Figure 11. Block diagram of arbitration

University of Hertfordshire

6.5.2 Buses

When a processor does not desire access to the shared memory it will be in the state of accessing local memory or remaining idle. When a processor is in either of these states its' bus signals must be isolated from the shared memory buses. If this was not done then each processors bus would always be connected to shared memory and the buses of the other processor. A situation like this would eliminate the purpose of arbitration and cause corruption of both processors buses resulting in a useless system.

Figure 12 shows a block diagram of the system including arbitration and the bus isolators.

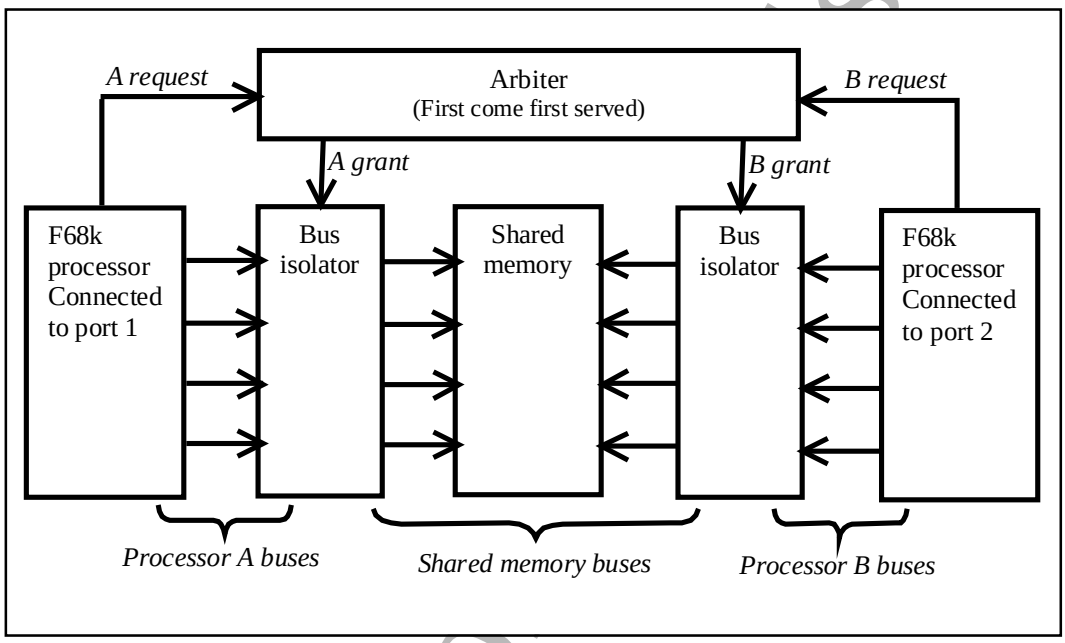


Figure 12. Shared memory module block diagram

University of Hertfordshire

7. Research

7.1 Literature search

A literature search was carried out using the online databases INSPEC and IEEE Xplore to search for relevant journal articles. Motorolas' web site was searched for datasheets relating to the MC68000 microprocessor and application notes that were relevant to the project. General searches were also conducted on the internet for any relevant information.

7.2 Hardware research

A search was conducted on the internet to see if any similar products were already in production, including the Flight Electronics web site. No similar products were found by Flight Electronics or any other manufacturer. The only results found were for industrial systems of which no design details were given, as might be expected for commercial reasons.

Detailed hardware design and implementation of information gained from the literature search starts in chapter 8.

7.3 Software research

Research was carried out into the requirements of the kernel. The kernel needs to demonstrate the functions of the shared memory module and allow others to explore them. Research was conducted into specific areas such as software arbitration including semaphores, inter-process communications including mailboxes, polling and interrupts and also into the C language and using pointers for direct access to a physical address space. Some research was also conducted on command line interpreters with the vision of implementing such a user interface as a gateway into the functions of the shared memory module.

More detail into the software requirements for the project is given in chapter 12.

University of Hertfordshire

8. Circuit Design

8.1 Address decoding

The F68k memory map is shown in Appendix A. Any unused (illegal) area of the memory map could be used as the location for shared memory with the address decoding being provided by appropriate logic. The 128*kByte* address range of A80000 to A9FFFF (extended) was chosen as it is towards the top of the memory map but retains enough memory space above to allow further expansion in the future.

When addressing an unused area of memory, the F68k generates a bus error signal and this needed to be disabled by removing link LK1 so that the unused address areas can be accessed.

It was decided to include the option of either extended or expanded memory on the module with expanded memory including two pages of 64*kBytes* and extended memory being the 128*kBytes* of the two pages combined. The user needs a method of switching between expanded or extended mode and switching between pages when in expanded mode. A software addressable output port was included in the circuit for this purpose and is described in chapter 8.3

Externally, the Motorola 68000 is seen as having a 23-bit address bus as it does not provide address line A_0 with the consequence that odd addresses can not be accessed. Instead the 68000 provides an upper and lower data strobe (UDS and LDS) which are used to select an upper or lower byte physical RAM chip. Therefore, the 64Kbytes of shared memory is split into an upper (8-bit) memory block and a lower (8-bit) memory block giving 64*Kbytes* or 32*Kwords* of shared RAM.

The number of address lines required per 32*Kbyte* RAM is given by:

$32Kbytes - 2^{10}.2^5$, summing powers gives $10+5-15$ address lines needed.

The number of address lines that need to be decoded is therefore given by $23-15=8$ for full address decoding.

The binary representation of the eight bits to be decoded is shown in Figure 13.

23	22	21	20	19	18	17	16	Address line	Space
1	0	1	0	1	0	0	1	A9FFFF	64*kbytes*
1	0	1	0	1	0	0	1	A90000	
1	0	1	0	1	0	0	0	A8FFFF	64*kbytes*
1	0	1	0	1	0	0	0	A80000	

Figure 13. Address decoding bits

From figure 5 it can be seen that the following Boolean logic expression is true between A80000 and A8FFFF (expanded memory).

$$Shared_access - A_{23}.\overline{A}_{22}.A_{21}.\overline{A}_{20}.A_{19}.\overline{A}_{18}.\overline{A}_{17}.\overline{A}_{16}$$

When extended memory is selected the address decoder needs to accept addresses between A80000 to A9FFFF. The Boolean logic expression for the address decoder when in extended memory is therefore:

$$Shared_access = A_{23}.\overline{A}_{22}.A_{21}.\overline{A}_{20}.A_{19}.\overline{A}_{18}.\overline{A}_{17}.(\overline{A}_{16} + A_{16})$$

University of Hertfordshire

This expression implies that A_{16} can be ignored. However, A_{16} needs to be an input to the decoder when in expanded mode to prevent access outside of the allocated address range. Therefore, to avoid having one decoder to recognise expanded mode and another to recognise extended mode, the input A_{16} can be forced low whenever extended memory is enabled. The decoder would then recognise extended memory and expanded memory as the same and grant access to shared memory.

A circuit to fulfil the above requirements for an address decoder is shown in Figure 15 where '~ext' is an active low extended mode signal which when asserted (logic low) ties the output of the AND gate, and hence A_{16}, low. When in expanded mode the output of the AND gate, and hence A_{16}, is unaffected. The truth table for this function is shown in Figure 14.

~ext	**A_{16}**	**Output (A_{16})**	**Description**
0	0	0	Extended mode, A_{16} forced low
0	1	0	Extended mode, A_{16} forced low
1	0	0	Not extended mode, AND gate output = A_{16}
1	1	1	Not extended mode, AND gate output = A_{16}

Figure 14. Logic to force A_{16} low when in extended mode

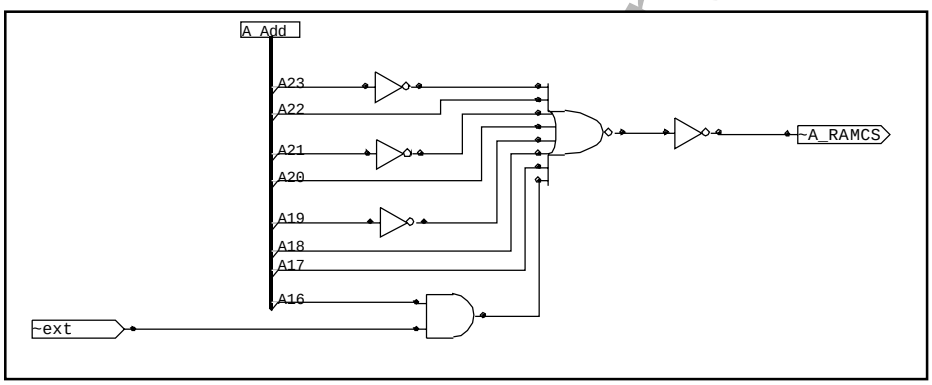


Figure 15. Address decoder circuit

The address decoder described so far is adequate for generating a 'shared memory request' signal. The actual RAM chip (upper or lower byte) can be selected by the data strobes however two pairs of RAM chips exist, with the second pair being extended memory or the second page of expanded memory. Therefore, the actual RAM chip select logic needs to take into account the memory mode and the page. The circuitry required to generate the RAM chip selection is best located after address decoding and arbitration so that it only needs to be produced once and not once for each port.

Three signals have been identified that need considering when selecting the physical RAM chip, these are UDS, LDS and PAGE. Although there is no second page when in extended memory a decoder can recognise the PAGE signal as an extended signal when in extended mode and the extended range is addressed.

A standard 74HC138 3-bit to 8-bit decoder is suitable for this purpose. The truth table for the decoder is given in Figure 16 (UDS and LDS are active low).

University of Hertfordshire

UDS	LDS	PAGE	74xx138 Output	Required operation	Description
0	0	0	Q_0	(A) Page 1 upper + lower	
0	0	1	Q_1	(B) Page 2 upper + lower	
0	1	0	Q_2	(C) Page 1 upper	
0	1	1	Q_3	(D) Page 2 upper	
1	0	0	Q_4	(E) Page 1 lower	
1	0	1	Q_5	(F) Page 2 lower	
1	1	0	Q_6	(G) Page 1 not selected	UDS and LDS disable
1	1	1	Q_7	(H) Page 2 not selected	RAM despite PAGE

Figure 16. RAM chip select logic truth table

The outputs of the chip select decoder also need combining to produce the required operation from the 74HC138 output. The following Boolean logic expression shows the combinational logic required to achieve this (RAM chip selects are active low):

$CSL_P1 = \overline{Q_0} + \overline{Q_4} = Q_0.Q_4$, Chip select, lower byte, page 1

$CSU_P1 = \overline{Q_0} + \overline{Q_2} = Q_0.Q_2$, Chip select, upper byte, page 1

$CSL_P2 - \overline{Q_1} + \overline{Q_5} - Q_1.Q_5$, Chip select, lower byte, page 2

$CSU_P2 = \overline{Q_1} + \overline{Q_3} = Q_1.Q_3$, Chip select, upper byte, page 2

The circuit that satisfies these expressions is shown in Figure 17.

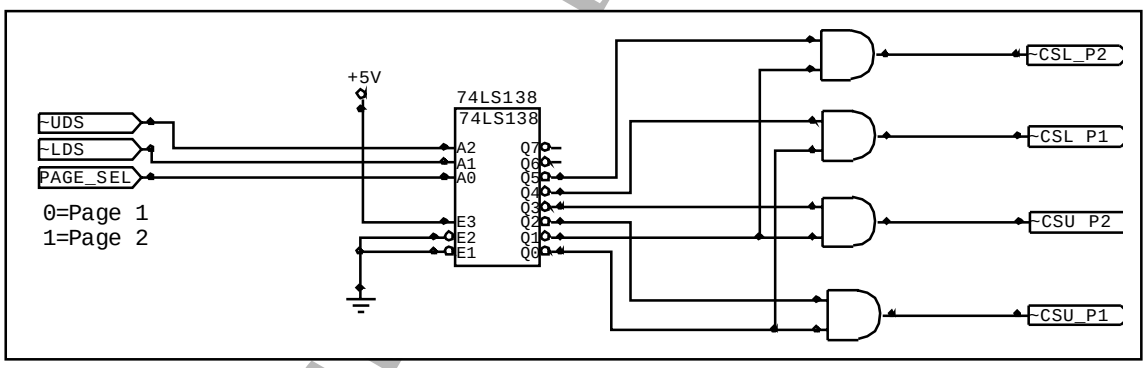


Figure 17. RAM chip select circuit diagram

University of Hertfordshire

8.2 Simulation of address decoding design

The inputs and outputs of the circuit shown in Figure 17 were simulated in Circuitmaker (SPICE). The waveform resulting from the simulation is shown in Figure 18 where the upper and lower byte outputs can be seen responding correctly to the upper and lower data strobes and the page select signal.

When UDS and LSD are low and PAGE is low, meaning page 1 is selected then U_1 (Upper byte page 1) and L_1 (Lower byte page 1) are also asserted low as expected.

Next PAGE goes high and therefore selects page 2 but UDS and LDS remain low. After a propagation delay U_1 and L_1 go high and U_2 (Upper byte page 2) and L_2 (Lower byte page 2) are asserted low, as expected.

The rest of the waveform shows that the upper and lower pages respond correctly to the inputs UDS, LDS and PAGE. If UDS is high then both upper byte pages are also high and if LDS is high then both lower byte pages are high.

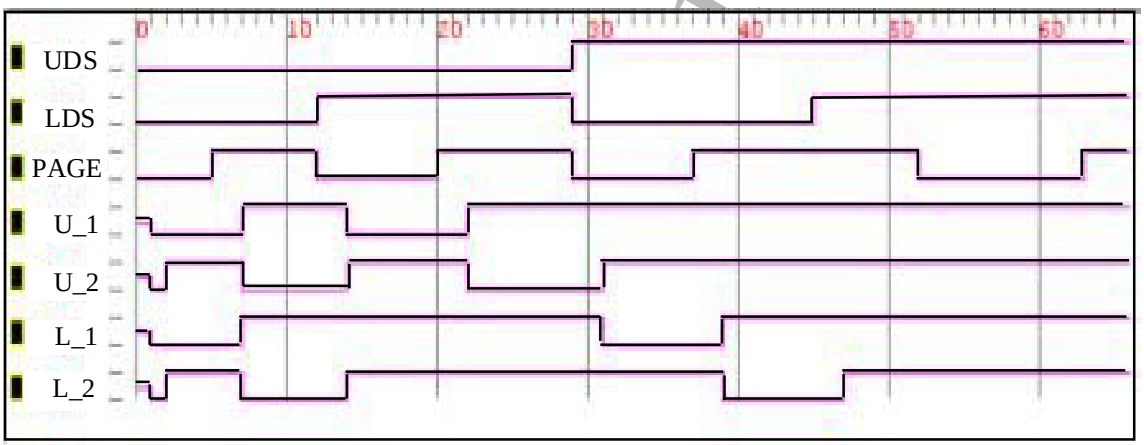


Figure 18. RAM chip select simulated waveform

8.3 Output latch

The PAGE signal described in chapter 8.1 is generated from the microprocessor via a memory mapped output latch on-board the shared memory module located at address FFFFFF because this is another unused space at the top of the memory map. Further address decoding is also needed to decode an access to the latch and the first eight bits of the data bus need to be connected to the latch to carry the output byte. A suitable latch is the 74HC373 since it is a readily available device and performs all the functions required.

To fully decode the address of the output latch all 23 bits of the address bus and the lower data strobe need decoding. The circuit shown in Figure 19 achieves this.

University of Hertfordshire

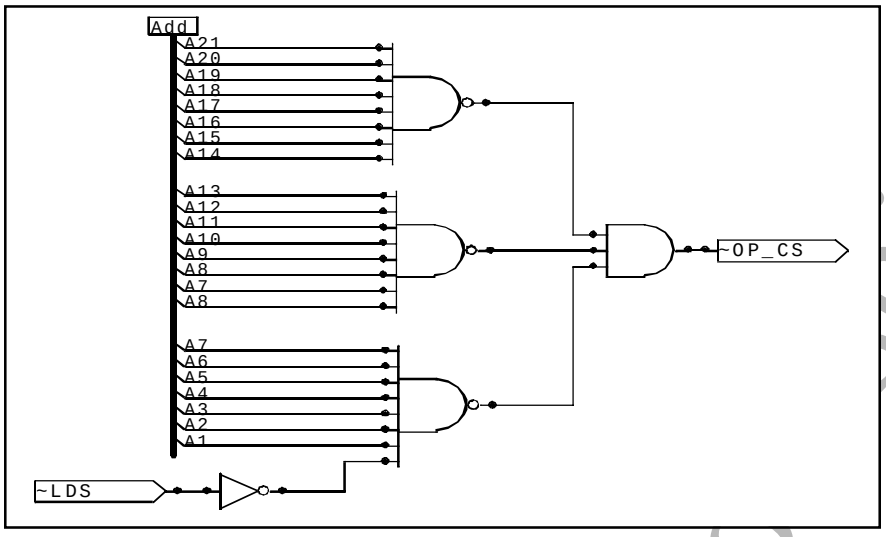


Figure 19. Output latch address decoding

The master microprocessor needs to be in control of memory configuration and therefore only the master must be allowed to write to the output latch, otherwise master and slave processors could contradict each other if poor code is ever written for them, possibly resulting in corruption of stored data. However, it is not known which port will be connected to the master processor during use. To solve this problem a D-Type flip flop (74HC74) is connected to each latch and when the D-Type is triggered by the address decoding shown in Figure 19 (after arbitration) the appropriate latch is enabled and no reset is provided other than a power down of the shared memory module. This effectively means that as far as the hardware is concerned, the first processor to write to the output latch is considered master from then on and granted access to the latch. Software could be written so that the master includes an initialisation routine in which the output latch is written to and the memory configured.

The outputs from the latch are controlled directly from the latch address decoder following arbitration, so that the latch is transparent when being written to and latched when a write is completed. The circuit for this is shown in Figure 20.

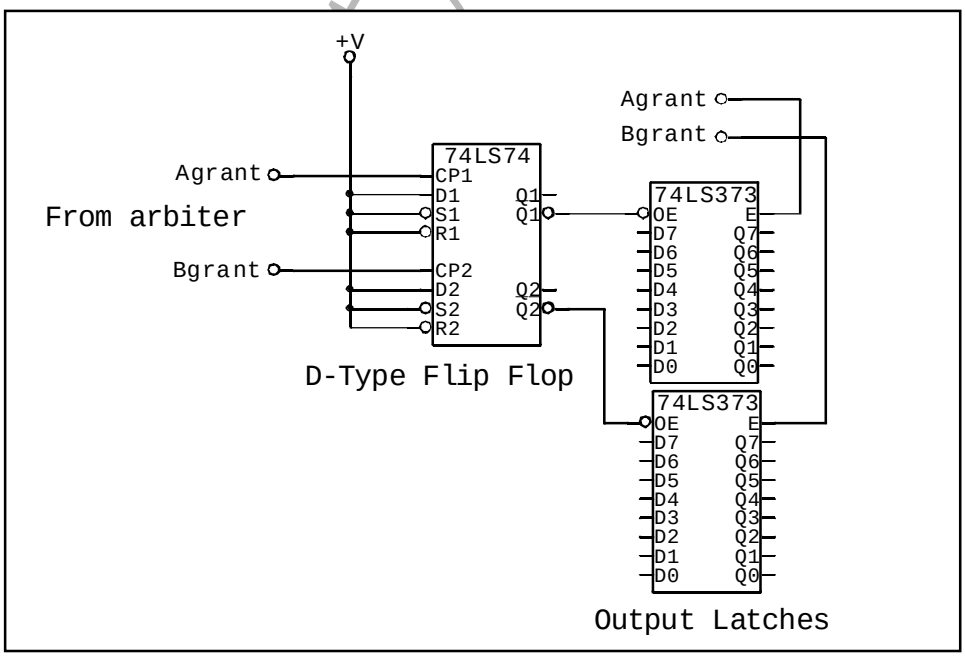


Figure 20. Output latch circuit

8.4 Arbitration

The arbitrator must be capable of accepting inputs (or requests) from the address decoding for RAM and the output latch for each microprocessor and therefore requires four inputs. The arbitrator has outputs corresponding to the inputs and must only enable one output at any one time. Access is granted following a request on a first come first serve basis because most other forms of arbitration (fixed priority, time shared, etc) could be implemented in software by the two processors communicating through shared memory.

Logically, a bi-stable circuit appears to provide the required functionality for an arbitrator, however if two inputs are asserted (make a request) at the same time then a bi-stable suffers from meta-stable race conditions where the outputs continually race back to the inputs and the outputs continue in an on-off loop.

Due to the meta-stable reasons mentioned, an off the shelf device from Philips was sourced. The 74F786 is a 4-bit arbitrator that is designed to eliminate the meta-stable race conditions and also operates on a first come first served basis and therefore fulfils the project requirements. This device was difficult to source as it is a new component and not listed in supplier catalogues but a search of Philips Semiconductors website revealed the datasheet.

The 74F786 also includes an AND gate which can be used to combine the grant signals and provide a signal meaning ‘an access has been granted’ but not specifying which one. This was used to trigger an LED which indicates memory access. A 74HC123 mono-stable was placed in the path of the LED to produce a one second, re-triggerable delay so that the LED is visible to the eye. The LED makes diagnostics easier and gives a visual indication of function to users of the shared memory module, which is important considering the intended educational use.

8.5 Bus signal isolation

A standard way of isolating bus signals is by using tri-state bus transceivers or buffers. The 74HC244 is a standard and readily available buffer and the 74HC245 a standard and readily available transceiver. The buffers are needed to isolate the address and control buses because these buses are unidirectional. The data buses are bidirectional and therefore require the use of transceivers where the direction can be switched by the control bus read/write (R/W) signal.

The ‘access granted’ output from the arbitrator is connected to the chip enable pins of each buffer and transceiver associated with the corresponding ‘request access’ input. When access has been granted the buffers and transceivers are seen only as a very low resistance but when access has not been granted and the buffers and transceivers are not enabled they are seen as a very high impedance and therefore do not interfere with other bus signals that may have been granted access.

University of Hertfordshire

8.6 DTACK and RAM

DTACK is also controlled in the same manner as other bus signals. When the address buffers are not enabled DTACK is also not asserted, but when the address buffers are enabled DTACK (active low) is asserted and the microprocessor recognises that access has been granted to it. In some system designs DTACK needs delaying. This is because the microprocessors in these systems can access the memory faster than the memory can operate.

Appendix N shows that the read and write cycles of the 68000 are a minimum of four clock periods long. The Flight 68000 uses a 10*MHz* clock, as such each clock cycle is 100*nSec* giving a minimum read or write time of $4*100nSec = 400nSec$. This means that the overall delay of the RAM and interface components including buffers, arbitration, etc must be less than 400*nSec*.

Appendix H shows the calculated access times of a read and a write cycle to access RAM, which is deduced from the circuit diagram in Appendix D plus the device datasheets. These calculations show that the maximum access time that a RAM device can have in order to work with the shared memory module is 318.4*nSec*. A readily available device was the HM62256-10T which is a 32*kbyte* static RAM with an access time of 100*nSec* and therefore fulfils the requirements.

8.7 Power supply

The shared memory module requires an external DC power supply providing +5Volts to the board. For simplicity it was decided to provide a +5Volt regulator, the LM7805, on board. This enables a voltage input within the range 7.5Volts to 35Volts and allows the 7.5Volt power supply that comes with the F68k to be used, or any other common 12Volt power pack.

Appendix I states the power requirements of the circuit. The calculated values are too high to work with the desired power supplies however the measured current was much less. It was therefore decided to use a 7.5Volt power supply and to fit a heat sink to the voltage regulator. Testing was conducted with this configuration without any problems being found. The DC power socket provided on the shared memory module also allows easy connection to a bench power supply with the appropriate connector.

A power indication LED was also included as a visual aid. Also, one decoupling capacitor (100*nF*) per integrated circuit was included to reduce the effects of noise. The capacitors were placed as close to each device as possible.

University of Hertfordshire

9. Printed Circuit Board and Mechanics

A major issue for the project that needed consideration was how the circuit should be constructed. It was decided that construction on breadboard would be too unreliable given the amount of portability needed between laboratories and also that crosstalk would become a problem with trailing wires between the shared memory module and the microprocessor board.

Another option was to use development board of a suitable size and with suitable holes for the connectors. The price and size of such development boards was examined and no suitable one was found. The decision was therefore made to design and construct a printed circuit board, which became an extra major task for the project plan.

The PCB was designed using Protel 99SE available in the Project Laboratory. The following issues had to be considered when designing the PCB:

- Board dimensions
- Board shape
- Board silkscreen (top/bottom layer print)
- Track sizes
- Hole sizes
- Position of devices
- Number of track sides
- Number of component sides

The board dimensions and shape determined the overall outline which was drawn manually within Protel to the desired measurements. The required width of the board was determined by placing two F68k boards adjacent to each other with the bus expansion connectors lining up one on top of the other. Figure 21 shows the measurement taken for the board width and also the intended orientation when mounting the board for use.

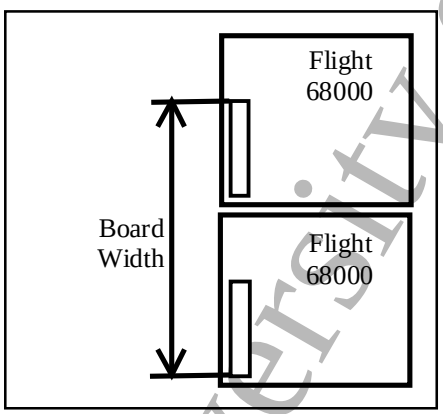


Figure 21. PCB width and correct orientation

The PCB height was kept to the minimum possible so that it was more portable and easily stored. The minimum height was determined by the amount of components on the shared memory module and their layout.

The components were placed on one side of the PCB because placing standard DIL components on both sides would not have saved any space and made design and trouble shooting harder. As components were only on one side, only the component side needed a printed silkscreen. The silkscreen was designed to show the device outline and device designation which made it easier to assemble the PCB. The words ‘power’ and ‘memory’ were also printed next to the appropriate LEDs to give a description of their function and the word ‘+12V’ was printed next to the power socket to indicate the maximum voltage that should be connected.

University of Hertfordshire

The shape of the board became rectangular because of the necessary width and final height, but a notch was also needed in order to clear a ribbon cable connector on the F68k. The ribbon cable connects the F68k to an application board and the shared memory module needs to clear this because it was perfectly feasible that the shared memory module be used in conjunction with the applications board during future use and during testing.

In order to place the components and route the tracks, the PCB netlist had to be imported into Protel from Circuitmaker. Once imported the netlist could be displayed so that the extent of connections could be seen. The routing of tracks would be an impossible task to complete manually on anything other than a very small circuit board and the netlist showed that this was not a small circuit. Therefore routing was completed automatically using Protels' autorouter function. The autorouter showed that the PCB needed to be double track sided due to the amount of connections. The vias (or through holes) were increased from the default size of 0.7mm to 1mm. Power tracks were increased to 2mm width and signal tracks were set to 1mm.

The devices were placed in position manually by visually looking for the shortest route for the displayed nets and aiming to keep associated devices together (such as all RAM devices). The power connector was located at the top edge of the board so that it was easily accessible and the LEDs were located next to it so that they were easily visible.

The completed PCB design was sent to a third party for manufacture (PCBPool) which took approximately two weeks to complete. Once the PCB was obtained the components were soldered in place. All integrated circuits were placed into DIL sockets for easy removal or replacement during testing, this is particularly important for a prototype.

Appendix F shows the PCB layout and measurements. Appendix M shows the Flight 68000 circuit diagram including the pin connections of the bus expansion connector on page 5. The bus connector is a 64pin (2 rows of 32) type Q right angled DIN41612. No decisions were needed in choosing this connector because it must mate with the bus expansion connector of the Flight 68000 board. The datasheet for this connector was obtained and the PCB footprint was drawn in Protel.

During assembly of the components onto the PCB it was noticed that the DIN41612 bus connectors sat too far away from the edge of the PCB. This resulted in the connector sitting at an angle due to a plastic rim on the edge of the connector. This was solved by shaving off the plastic rim, with no adverse affects.

It was noticed during the PCB design stage that the auto routed tracks for each ICs decoupling capacitor did not always connect directly to the power pins of the IC they were located next to. Attempts were made to correct this but any manual modifications affected those tracks that had been automatically routed. The decision was made to leave the tracks routed automatically and solder wires directly between the decoupling capacitors and the IC power pins after manufacture.

University of Hertfordshire

10. Power Test Results and Debugging

10.1 Methodology for power tests

The following tests were conducted after all IC sockets and other components had been soldered in place but ICs themselves were removed.

1. Visual inspection
2. Voltage measurement of supply
3. Current measurement of supply
4. Voltage measurement of IC pins
5. Voltage measurement of all bus connector pins

The steps taken during testing and the results of these tests are detailed below.

10.2 Results and rectifications for power tests

10.2.1 Visual inspection

The visual inspections involved looking for dry joints, unsoldered or poorly soldered joints, solder splashes causing a short circuit and incorrectly orientated components.

10.2.2 Voltage and current measurements

The voltage and current measurement of the supply was carried out by connecting to a bench power supply with digital displays of those quantities. The bench power supply also has current limiting so this avoided any problems that would have been encountered if the PCB had a major fault such as a short circuit between power lines.

The calculated and measured power values are given in Appendix I.

10.2.3 Voltage at IC pins

A voltage measurement was taken between the ground and power pins of each IC socket to test for the correct supply voltage (approximately +5V) as a means of identifying any open circuits.

A fault was discovered at this stage where some IC positions did not have any voltage across the power pins but did have a voltage to the positive pin when referenced to the power supply zero volts. This indicated an open circuit on the zero volt line. Resistance measurements were taken with the circuit powered down and the open circuit identified. The cause of this fault was examined and found to be an error in which the netlist contained two separate ground nets which were not connected together.

The fault was fixed by soldering a link between appropriate points on the two ground nets. No further faults were found at this stage.

10.2.4 Voltage on bus connector

Measuring the voltages on the bus connector pins was a very important test because any power connected the wrong bus signal could have potentially damaged the F68k microprocessor board.

No voltages were measured on any bus pins and therefore no faults were found at this stage.

University of Hertfordshire

11. Functional Test Results and Debugging

11.1 Methodology for functional tests

Following successful tests and rectification of the above power tests, functional testing was carried out. Functional testing involved inserting all ICs into the circuit and connecting the shared memory module to one or two Flight 68000 microprocessor boards as the individual tests required.

The following functional tests were conducted.

1. Memory write
2. Memory read
3. Address bus walking 1s test
4. Data bus walking 1s test
5. SISD test
6. SIMD test
7. MISD test
8. MIMD test

The steps taken during testing and the results of these tests are detailed below.

11.2 Results and rectifications for functional tests

11.2.1 Memory read and write

The initial tests for memory read and write were carried out by writing a C program with a pointer to the shared memory address range starting at A80000. The C code used for this is shown in Figure 22 which was entered using the Crossware C compiler of the Embedded Development Studio (EDS).

```
#include <stdio.h>

void main()
{
   int *mem;
   mem = 0xA80000;    // A80000 is first shared memory location
   *mem = 0xAAAA;     // Place AAAA (alternating 1s at shared location)

   printf("Address %d contains %d",mem,*mem);
}
```

Figure 22. Initial C test program

The results of running this program were shown in the EDS output window, with the returned address and data values that the 'printf' statement displayed as an indication of correctness. The test was carried out with each port, one at a time.

Both ports returned similar results during this test but neither functioned correctly. The values returned were not as expected and each time the program ran a different value was returned.

In order to bring testing down to a lower level and eliminate the C program, compiler and linker from the investigation of the fault, the F68k was fitted with the original monitor EPROMs providing the command line user interface.

University of Hertfordshire

With the monitor installed, the 'MO' command was used to read and write values to specific memory locations. The equivalent monitor command of the C program in Figure 22 is shown in Figure 23, where *XXXX* is the initial returned value, AAAA is the hexadecimal value to be written and *ZZZZ* is the returned value after the initial write. If the first write was successful and AAAA was entered into memory then *ZZZZ* would equal AAAA.

```
F> MO
   ENTER ADDRESS : A80000
00A80000      XXXX   AAAA

F> MO
   ENTER ADDRESS : A80000
00A80000      ZZZZ
```

Figure 23. Monitor terminal window showing read/write command

The results obtained from this test showed the following symptoms of a fault.

- Some addresses did not respond to the monitor
- Some addresses appeared stuck at a certain data value
- Some addresses sometimes read the correct value and sometimes the incorrect value

These results were similar to the results from the C program as arbitrary values were being obtained. This led to the visual inspection and continuity tests of the address and data lines on the shared memory module PCB.

Several address lines were found to be missing from the PCB tracks as detailed below.

1. Upper byte page 1, address line A2 had no track on the PCB.

   This was caused by a circuit diagram mistake. Although the wire had been drawn, it was missing from the netlist because one end of the wire had been labelled as 'D2' instead of 'A2'. Instead of flagging a possible error when exporting the netlist, Circuitmaker had left off the connection.

   This also meant that every other RAM device had an open circuit A2 because of the way in which the PCB tracks were routed. This would mean overlapping addresses within the memory devices.

   The fault was rectified by soldering small solid core wire-wrap wire between the correct points.

2. Port 2 address lines open circuit.

   The visual inspection also showed the same fault as in fault 1 above, but with all the address lines between the RAM devices and port 2 buffers open circuit.

   This was rectified by soldering wires between all the necessary points.

Following the rectification of these two faults the read/write tests were repeated in the same manner. The results differed from those being obtained before but were still not correct. Figure 24 shows the results following rectification of the address bus.

The error message '***NO CHANGE***' was displayed after a value was entered although the upper byte of the returned value was always changing to the inputted value. This pointed to a

University of Hertfordshire

problem with the lower byte RAM chip selection and a visual examination of the data strobe pins on the bus connectors showed that the lower data strobe (LDS) had no track going to it. A study of the circuit diagram showed that this connection had again been labelled incorrectly and was missing from the netlist.

```
F> MO
    ENTER ADDRESS : A80000
00A80000      XXXX  AAAA
  XXAA ***NO CHANGE***
```

Figure 24. Monitor terminal window following address bus corrections

The missing UDS was found on both ports and was rectified by soldering a wire between the correct pins on the PCB. The read/write tests were then repeated on both ports, however similar results were obtained, with the fixing of the UDS signal making no apparent improvement.

In an attempt to prove that shared memory was being accessed, the RAM devices were removed from the circuit and the read/write tests repeated. The returned values were random as they had been with the RAM devices installed which proved that shared memory was not being accessed at all.

In order for the 68000 microprocessor to complete a bus transaction and therefore for the Flight monitor to complete an MO command it must receive back DTACK. For this reason the buffer on the shared memory module that was responsible for generating DTACK was removed from the circuit so that the effects could be observed. The read/write test was repeated with no difference to the results. This indicated that the F68k was generating a DTACK signal itself otherwise the MO command would have sat in a loop waiting for DTACK.

Another test to reinforce this was done by removing the shared memory module completely and repeating the read/write test. Again there was no difference in results. To prove that the F68k was generating DTACK, a logic analyser was obtained and connected to the 68000 microprocessor. DTACK was observed being asserted at regular intervals both when a read/write was initiated and when no read/write was initiated with the MO command.

The results obtained up to this point indicated that although the F68k manual and address map claimed that A80000 to A9FFFF were unused areas, they were not. The MO command was then used to open addresses within the range FF0000 upwards. Several addresses were tried and the MO command stayed in a loop until the F68k was reset, as it should have done with only the current A80000 to A9FFFF address range being decoded. This indicated that when FF0000 and above was addressed, DTACK was not being generated and was therefore an unused area as the manual and memory map also stated.

The conclusion was reached that the address range A80000 to A9FFFF where shared memory was mapped in to, was in use by the F68k and therefore the shared memory module had to be mapped in to a different location of the address map. As FF0000 and above had been tested and determined unused, the address decoding of the shared memory module needed modifying to shift shared memory up the address map to FF0000.

The new address decoding needed to decode the highest eight bits of the address bus (A_{23} to A_{16}) with the following Boolean expression

$$\overline{memory_access} = A_{23}.A_{22}.A_{21}.A_{20}.A_{19}.A_{18}.A_{17}.A_{16}$$

University of Hertfordshire

This expression required an eight input NAND gate which was not pin compatible with the eight input OR gate already in place. The address decoding for the output latch did however contain eight input NAND gates (U1, 2, 3, 8, 9 and 10) and with some modifications to the PCB tracks by soldering wires, one of these per port could be used instead. This meant however, that the output latch would have no address decoding which meant that the output latch function had to be sacrificed and this in turn meant that the choice of extended or expanded modes was sacrificed. This option was still chosen because it was the simplest choice given the time that remained.

Another issue that had to be considered and was fortunately known before this modification was done is that the 68000 microprocessor places its address bus high when generating an interrupt. This would have been interpreted by the shared memory module address decoding as a valid request for shared memory which would have opened the shared memory bus to the microprocessor bus thereby corrupting the interrupt data.

To avoid the problem of invalid shared memory access during interrupts, an interrupt acknowledge signal needed generating. This was done by recognising the 68000 function code outputs as $FC0.FC1.FC2 = interrupt$ through an AND gate. This signal also had to be combined with the address strobe signal which signifies the function codes being valid. The interrupt acknowledge signal was combined with the address decoding output as shown in Figure 25.

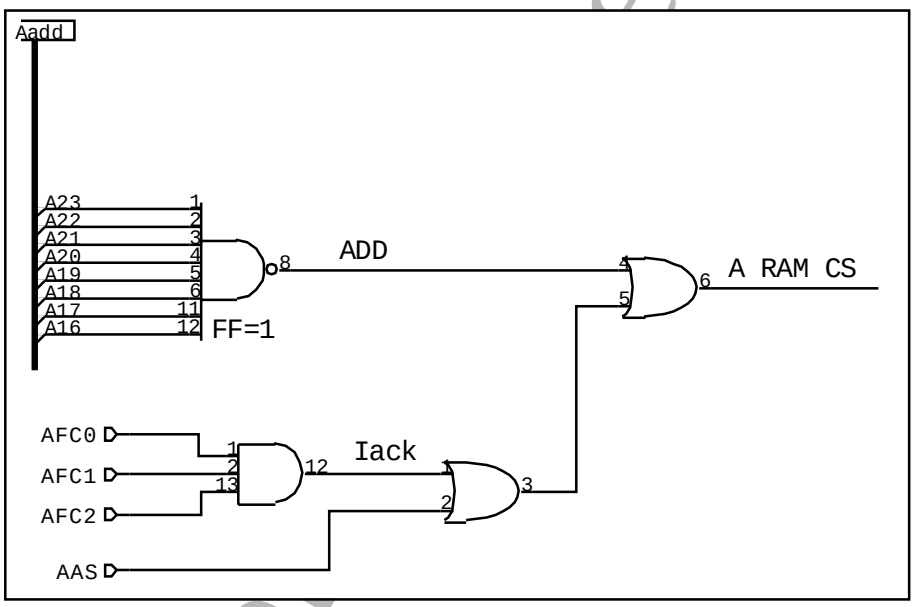


Figure 25. Modified address decoding

This modification involved adding one AND gate (U37) and two OR gates (U36) per port to the circuit. These devices were not present on the PCB and so a stripboard circuit was constructed and wired to the PCB with the stripboard being insulated by a paper backing and taped to the back of the PCB so that it remained reliable and portable. This modification is shown in Figure 26.

University of Hertfordshire

Figure 26. SMM under test showing modification

After the address decoding was changed and the interrupt acknowledge modification in place the read/write test shown in Figure 24 was repeated. The 'MO' command was entered as before but did not return an initial value. This showed that the command was sitting in a loop, which was normal if it had not received DTACK. The logic analyser was then connected to the buffers responsible for generating DTACK on the shared memory module. The logic analyser showed that the DTACK signal was at logic high and did not change to a low state when a read/write was initiated by the 'MO' command. A visual inspection of the PCB then revealed that no track connected between the buffer and the bus connector and was therefore open circuit. The circuit diagram was checked and as previously found with missing bus signals DTACK had been numbered incorrectly which had caused the connection to be missing from the netlist. This was solved by soldering wires in place for both ports.

The read/write shown in Figure 24 was repeated again following the DTACK signal rectifications. The results obtained were exactly as the previous results when DTACK was open circuit. The symptoms no longer indicated an obvious fault, other than DTACK remaining open circuit, but DTACK was verified and proved to be correct. Further analysis with the logic analyser was conducted by tracing through the system from the beginning (address decoding).

The logic analyser was placed on U1 and the inputs and output to this device were monitored when a read/write was initiated. Figure 27 shows the upper eight bits of the address bus used for decoding a request for shared memory. The address bus is shown changing state during normal accesses to local memory prior to and following the read/write to shared memory. The request signal (U1 output) is shown changing to a logic low state when all inputs are logic high during the read/write. This proved that the address decoding was performing correctly. This test was repeated for port two (U8) with the same results.

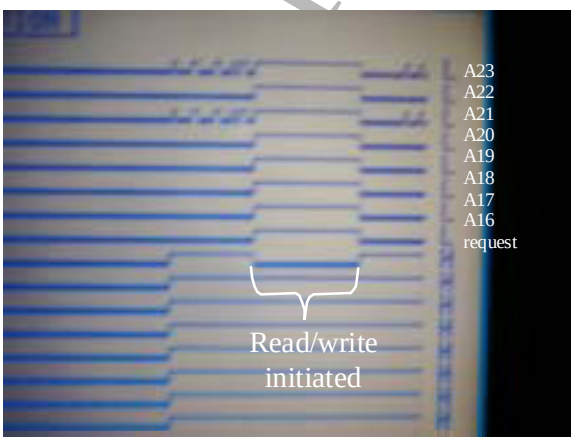


Figure 27. Address decoding waveform

rsity of Hertfordshire

The next stage in the circuit to be examined with the logic analyser was the combined output from the address decoding and the interrupt acknowledge (U35B output). This was measured as a logic low when a read/write was initiated which was a valid active low request for shared memory to the arbiter. The shared memory module was swapped around so that the second port could be tested, this produced the same results.

Arbitration was the next stage in the circuit and the logic analyser was connected to the bus arbiter U13. The request input to the arbiter from the port currently connected was active at logic low but the request input from the second port, which was not connected, was also present at logic low, as was its' corresponding output, which meant that access had been granted by the arbiter to the disconnected port.

The second ports request signal was then traced backwards through the circuit to the address decoding and with the second port still disconnected the inputs and outputs were observed. The logic analyser showed that the address decoding output was low and therefore producing a request signal. It became clear that this was because when either port was disconnected, the inputs to the address decoding for that port were floating. These floating inputs were being interpreted as logic high by the address decoding logic and since the earlier modification had moved the shared memory address range to FF0000, the high floating inputs were seen as FF and therefore a request for shared memory. This request from the disconnected port was being granted access by the arbiter because the request was being generated at power up (first come first served) whereas the connected port had a delay before the read/write was initiated by a typed command.

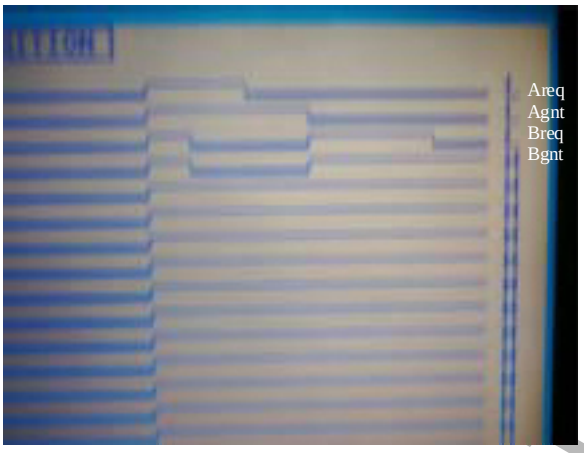
A second F68k was obtained and the shared memory module was connected to two processors at the same time, for the first time during testing. The read/write test was repeated and showed correct results, which was proved by repeating the test shown in Figure 23 where *ZZZZ* equalled AAAA.

The read/write test was repeated several times and at several memory locations. The results obtained from the test became inconsistent and after disconnecting and reconnecting the shared memory module in an attempt to ensure a good connection, the read/write test constantly returned incorrect values for the data in memory or did not return any values and instead acted as though DTACK had not been received. It was noticed that when a finger was placed over the interrupt acknowledge logic devices, that the read/write test returned values when it had otherwise been sitting in a loop waiting for DTACK to be asserted. A logic analyser and visual investigation of this fault revealed a missing wire link on the stripboard containing the interrupt acknowledge components. This link was inserted and read/write tests repeated with correct results obtained. Logically, it would be expected that without the missing link, the circuit would not function. The reason correct results were originally obtained even with the link missing can be explained by crosstalk acting in a sometimes constructive, but inconsistent manner.

University of Hertfordshire

11.2.2 Arbitration test

The logic analyser was connected to the bus arbiter (U13) and the waveform produced is shown in Figure 28. The waveform first shows that neither port is requesting access and therefore neither port is granted access. Port 2 then requests access and since there is no contention from port 1, port 2 is immediately granted access. Before port 2 has completed its bus cycle and removed its request port 1 has requested access. The waveform shows that port 1 is not granted access until port 2 removes its request. Finally the waveform shows that whilst port 1 is using the bus port 2 requests access which is not granted because port 1 currently has access.



Where:
Areq = Port 1 request
Agnt = Port 1 grant
Breq = Port 2 request
Bgnt = Port 2 grant

Figure 28. Bus arbitration waveform

11.2.3 Address bus walking 1s test

A walking 1s test checks that each individual address line functions correctly by ensuring no open circuits or short circuits between address lines. The test is carried out by monitoring the address bus with a logic analyser and performing a read or write to memory locations that cause the binary value of that address to increment by one address line each time, with all others at logic low. The table in Figure 29 shows the address values and the address line being tested. Only the address lines used for actual addressing needed to be tested using this test because the upper eight decoding address lines (A_{23} to A_{16}) were tested previously. Also, note the absence of address line A_0 as this is not an external address line.

University of Hertfordshire

	Binary address bus lines														
Address	2^{32768}	2^{16384}	2^{8192}	2^{4096}	2^{2048}	2^{1024}	2^{512}	2^{256}	2^{128}	2^{64}	2^{32}	2^{16}	2^{8}	2^{4}	2^{2}
(Hex)	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
4	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
8	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0
10	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
20	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
40	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
80	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
100	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
200	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
400	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
800	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
1000	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
2000	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
4000	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
8000	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29. Table to show values used for walking 1s test

This test was conducted after the previous repairs of missing PCB tracks and the results obtained proved that the address bus signals were present and correct. The diagrams in Appendix K show the waveforms produced on the logic analyser against each hexadecimal address.

11.2.4 Data bus walking 1s test

A walking 1s test was also carried out on the data bus in exactly the same manner as the address bus. The only difference being that the data bus tests included the value 1, because the least significant data line, D_0, is present, unlike the least significant address line, A_0. The test proved that the data bus lines were present and correct. The waveforms produced from this test are shown in Appendix L against the hexadecimal data value.

11.2.5 SISD test

Proving that a Single Instruction Single Data architecture operates successfully with the shared memory module is a matter of verifying that just one processor can access and transact with shared memory. This arrangement was already verified by the tests conducted throughout the functional testing stage and therefore no further tests were necessary.

11.2.6 SIMD test

The Single Instruction Multiple Data architecture was tested by using an assembly language program written by the project supervisor and assembled using the F68k monitor. The test program was designed to produce a counting sequence on the LEDS of an F68k application board.

The aim of this test was to place (assemble) the test program into shared memory and run the program twice, once from each processor, at the same time. In this way, two version of the same program (Single Instruction) would be seen to run on the LEDs of two different processors (Multiple Data). The two programs would be offset from one another by the time difference between starting each program and the amount of time the first access to the arbiter lasts for.

University of Hertfordshire

The test program used is shown in Figure 30 and the subroutine for that program in Figure 31.

```
MOVE.B #$03,D1              ; Data register 1
MOVEA.L #$80000F,A1         ; PI/T control register
MOVE.B #$80,(A1)            ; Address register 1
MOVEA.L #$800007,A1         ; PI/T data direction register
MOVE.B #$FF,(A1)
MOVEA.L #$800007,A1
MOVE.B #$FF,(A1)
MOVEA.L #$800013,A1         ; PI/T data register
MOVE.B D1,(A1)
BSR.L $FF0600               ; Subroutine in Figure 31
ADD.W #$00001,D1
BRA.L $FF0028
ORI.B #$00,D0
END
```

Figure 30. SIMD test program

```
MOVE.W #$FFFF,D4
DBEQ D4,$FF0604
RTS
ORI.B #$00,D0
END
```

Figure 31. SIMD test program subroutine

When the test program was assembled and ran, the monitor returned the error message '*** OPCODE 1111 EMULATION' followed by the contents of the data and address registers dumped to the screen. The code was then disassembled in order to confirm correct entry and the disassembled code is shown in Figure 32. The first two instructions at addresses FF0000 and FF0002 are shown corrupted. The program was reassembled with similar faults occurring.

As the address and data buses had previously been verified, no immediate cause as to this fault was apparent. The corrupted lines of code were then entered individually by using the assembler to branch to the specific corrupted lines and entering only those lines. Several attempts at entry were made and the program was then run successfully from one processor.

The second stage was to run the same program from a second processor whilst the first processor remained running. This was also done successfully and both processors were observed running their own instances of the same program. This proved that SIMD was working with the shared memory module bringing two processors together in a parallel architecture with arbitration and bus isolators working as expected. However, after a period of time the second processor to run stopped running when it should have been in a continuous loop. The processor was run again successfully but crashed again. It was possible to restart the program on all subsequent retries but after an unspecified time it stopped running. This fault remains present as this stage of testing was carried out close to the project deadline.

University of H

```
F> DS
BLOCK START...
   ENTER ADDRESS : FF0000
BLOCK END...
   ENTER ADDRESS : FF003E
FF0000 0000              ????
FF0002 FFFF              ????
FF0004 227C0080000F      MOVEA.L #$0080000F,A1
FF000A 12BC0080          MOVE.B  #$80,(A1)
FF000E 227C00800007      MOVEA.L #$00800007,A1
FF0014 12BC00FF          MOVE.B  #$FF,(A1)
FF0018 227C00800007      MOVEA.L #$00800007,A1
FF001E 12BC00FF          MOVE.B  #$FF,(A1)
FF0022 227C08000013      MOVEA.L #$08000013,A1
FF0028 1281              MOVE.B  D1,(A1)
FF002A 610005D4          BSR.L   $FF0600
FF002E D27C0001          ADD.W   #$0001,D1
FF0032 6000FFF4          BRA.L   $FF0028
FF0036 00000000          ORI.B   #$00,D0
FF003A FFFD              ????
FF003C FFFE              ????
FF003E FFFF              ????
```

Figure 32. Disassembled test program

11.2.7 MISD test

Verifying operation of a Multiple Instruction Single Data architecture entails the same use of shared memory by the two processor as with SIMD. The processor would be competing for access to shared memory and therefore relying on arbitration but with MISD tests the processor would be gaining access to shared memory to retrieve data as opposed to instructions as with SIMD, but the test would prove nothing more than was proved with SIMD. Separate tests for MISD were therefore not conducted as SIMD tests were deemed to satisfy all aspects of MISD.

11.2.8 MIMD test

Multiple Instructions Multiple Data is satisfied by the correct operation of any processors that can communicate with one another and therefore function as a parallel architecture. The initial functional tests proved that shared memory could be written to and read from, including writing data from one processor and reading that data from the second processor. The read/write tests are a proof of MIMD since both processors were communicating.

MIMD is also apparent through the SIMD tests because proof was obtained of a working parallel architecture. Therefore no specific MIMD tests were conducted.

11.2.9 C Compiler

With the hardware largely proved working, the monitor EPROMs were swapped for the C compiler EPROMs in an attempt to repeat the read/write tests with the C program that was originally written and shown in Figure 22.

The compiler returned the error message shown in Figure 33 and proceeded no further. All subsequent attempts with this and different C programs produced the same error. Unplugging the shared memory module and using the compiler produced correct results (using local memory). The cause of this fault remains unknown due to time constraints but may be related to the problem described in chapter 11.2.6.

University of H

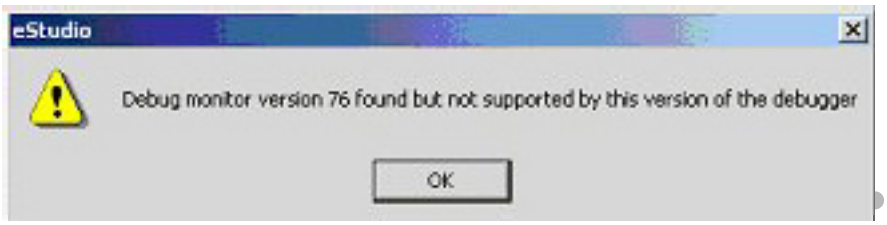


Figure 33. Compiler error message

University of Hertfordshire

12. Software

The project aimed to produce a kernel suitable for use in conjunction with the shared memory module hardware. The kernel was to provide a basic interface and enable a student to demonstrate the key features of shared memory and see a demonstration of Flynn's Taxonomy.

Research was conducted into software for use on parallel architectures, although time constraints on the project due to extended debugging of the hardware meant that no software development has yet taken place other than that used for testing purposes.

The areas of software that were researched include:

- Multiprocessing (multi-tasking)
- Software arbitration
  - Semaphores
  - Round robin
- Inter process communications
  - Mailboxes
- Command line interpreters
- Master / Slave configuration
- Initialisation of the shared memory module
- Use of memory paging (expanded memory)

The simplest implementation of a kernel would be one that is loaded into RAM by the user when required. The users' application would then be stored in a separate area of RAM with the ability to call functions within the kernel. Such functions would include system calls which the kernel would then perform through the F68k monitor. The kernel would also be responsible for memory configuration of different pages in extended mode and ensuring that communication between the two F68k processors took place correctly.

If a portion of shared memory was allocated as a mailbox, with a division for each processor then the kernel would also be responsible for passing messages into and reading from the correct mailbox. In this way, the user would not need to be concerned with specific addresses, but could simply pass a message via a kernel function and wait for a response as appropriate to the specific application.

Figure 34 shows a proposed hierarchy of the dual processor F68k system with shared memory and kernel.

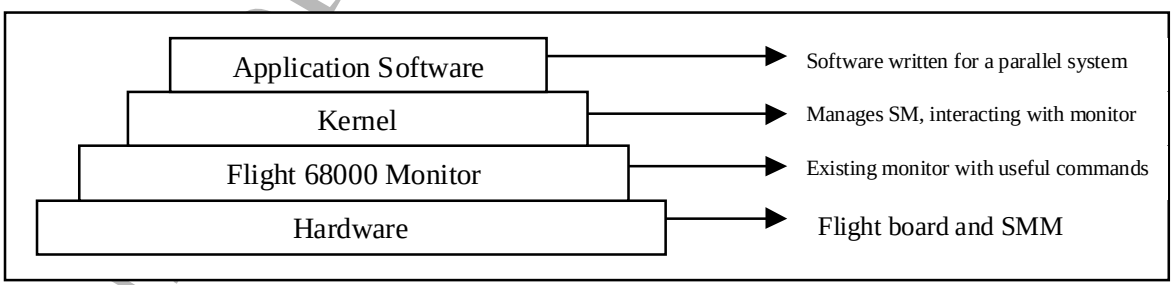


Figure 34. Software hierarchy

University of Hertfordshire

13. Conclusions

The projects' main objective was to develop a prototype shared memory module. The shared memory module was to act as a bridge between two Flight 68000 (F68k) microprocessor training systems, thus becoming a parallel architecture with the tightest form of coupling.

The parallel architecture resulting from the integration of the shared memory module and two F68k processors is intended for use within the microprocessor laboratory at the University of Hertfordshire as an educational aid.

The very first stages of the project were to research the requirements of the shared memory module by gathering information about the Motorola 68000 microprocessor and the Flight 68000 training system. Research was also conducted into software requirements of parallel systems and applications including Flynn's Taxonomy.

The different functions that were needed to fulfil the design requirements were identified and included arbitration and bus isolation. Research provided an 'off the shelf' solution for the bus arbitration which saved development time. The arbitrator and bus isolation was then included in the detailed circuit design.

The initial circuit design was constructed on printed circuit board (PCB) because of the size of the circuit and the reliability that was needed. The PCB was designed with the computer aided design package Protel which involved a steep learning curve and became an extra task within the project. The extra time and learning involved in producing the PCB was however very worth while, because the circuit is now available in a robust form that is available for future use and development if required.

Following circuit design and manufacture of the PCB, testing was conducted to test for conformance to the design. The initial stages of testing highlighted various missing tracks on the PCB which were due to labelling errors on the circuit diagram. The missing tracks were easily repaired by wire links.

Testing also revealed an apparent error in the manual for the F68k. Shared memory was designed to be located at an unused area of the F68ks' memory map, which was given in the F68k manual. However, testing revealed that the area used was not in fact unused, although the work carried out during the project did not, nor did it need to identify what did occupy that address space. The consequence of this was a redesign to the circuits address decoding and a change to the shared memory location. This unfortunately meant a sacrifice of the expanded and extended memory functions of the shared memory module because of modifications to the PCB.

Following modifications, the circuit was tested and proved to function correctly, according to the design. Evidence still exists of one fault which causes a processor to abort, but the passing of the tests that were conducted prove the circuit design is logically correct. The remaining fault would probably be small and not be too difficult to trace and repair, this was not done during the course of the project due to time constraints.

It is therefore perfectly feasible that with a little more attention to the one remaining fault and a consideration of the improvements mentioned in chapter 14 that the same overall design can be used and further shared memory modules manufactured and used within the laboratory as intended.

University of Hertfordshire

14. Further development

The shared memory module manufactured during this project was a prototype and as such there are many improvements that should be considered before any further units are produced.

14.1 PCB improvement

The PCB would need to be redesigned to accommodate the modifications that took place as a result of test and debugging. An improved physical layout of the PCB could also be produced where components are located in number order. This would make components easier to find when tracing faults.

The notch that was incorporated into the PCB shape to clear the ribbon cable connecting application boards to the F68k ideally needs a larger clearance to make insertion of the shared memory module easier. An increase from 10mm to 15mm would be appropriate.

The footprint for the bus connectors (P1 and P2) needs to be moved closer to the edge of the circuit board. A small problem was found when assembling that meant that a plastic rim on the connectors did not overhang the PCB but instead caused the connector to sit at an angle. This was easily solved however by shaving off the plastic rim from the connectors.

14.2 Circuit improvement

A significant amount of PCB space could be saved by incorporating much of the circuits' logic into a programmable logic array (PAL). Address decoding and chip select logic could certainly be included inside a PAL and possibly bus isolating buffers if a PAL with high impedance output capability was used. A PAL would also facilitate future modifications or corrections more easily.

14.3 Software

The intended kernel could be produced in the future as part of another project. The kernel could take different forms depending upon the depth of any such project. The level intended for this project was to write a kernel that was loaded into RAM when required and provided a command line. A more advanced kernel might include integration with the F68ks' monitor. Integration with the monitor would mean obtaining the monitor code listing and editing the EPROMs on which it is stored. This would however provide the tidiest form of the kernel. Any project undertaking software development for the shared memory module should take into consideration chapter 12.

University of Hertfordshire

15. Project management

15.1 Resources

The feasibility study conducted at the beginning of the project did not foresee any resource problems and no major issues were encountered during the project. At the beginning it was sometimes difficult to obtain access to the microprocessor laboratory and when access was gained test equipment had to be arranged. Making modifications to the circuit was also difficult within the microprocessor laboratory and often meant moving to the projects laboratory to make minor modifications and then back again to test the results. Equipment also had to be put away and could not be left out from one test session to another.

The possibility of obtaining F68k processors inside the projects laboratory was investigated but found not to be possible. If this had been possible then testing and debugging would have moved at a greater rate and been completed quicker and more efficiently.

The use of a logic analyser was not considered within the feasibility study, but was found to be needed and was subsequently obtained. There was approximately a one to two week delay for this.

15.2 Cost analysis

The parts list containing all parts used both before and after modifications to the circuit is contained in Appendix G. The total cost of all parts came to £160.96 of which a large proportion was for the PCB. Although the project had an initial budget of £50, no claims were made in the feasibility study to stick to this budget and all spending was authorised. There were therefore no budget problems with the project.

Given that the eventual prototype did not include the extended/expanded memory functions originally included in the design, it is reasonable to expect that a PCB designed without those functions would be smaller and cheaper, as might a PCB designed with a PAL replacing logic devices.

Total project time including development, assembly, test and rectification is estimated at 306 man hours. The total number of meetings with the project supervisor, including theory and practical is estimated at 10.

15.3 Time analysis

Chapter 15.4 shows the initial Gantt chart created at the start of the project. Efforts were made to adhere to the initial Gantt chart throughout the project but where this became impossible the Gantt chart was updated. The final Gantt chart at the end of the project, following all updates is shown in chapter 15.5.

A comparison between the initial and the final Gantt charts shows that the majority of tasks were completed on time. Component ordering was not completed by the time stated in the initial Gantt chart but this was due to further components being ordered for modifications during the testing phase and had no knock-on effect with any other tasks.

The two main tasks that were not completed at their initial deadlines were programming and testing. The start of testing was delayed due to the PCB having to be designed and ordered before it could be tested. Assembly of the circuit did not take any longer than initially planned for but it was pushed backwards, also due to the design and ordering of the PCB.

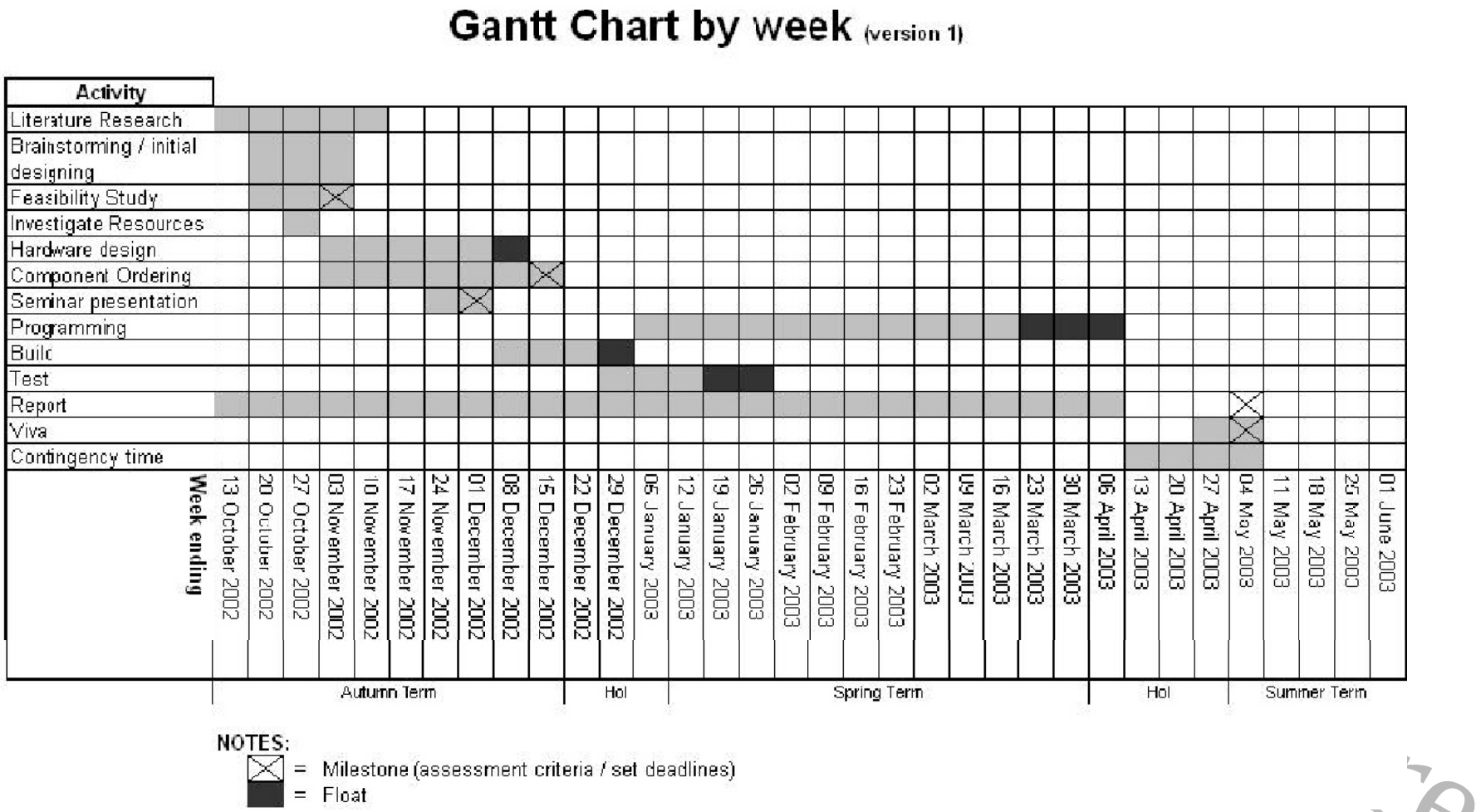
Testing was the task that had the main affect on the time plan. A large number of errors were found during testing which meant a lot of fault finding and rectification had to be done. The

University of Hertfordshire

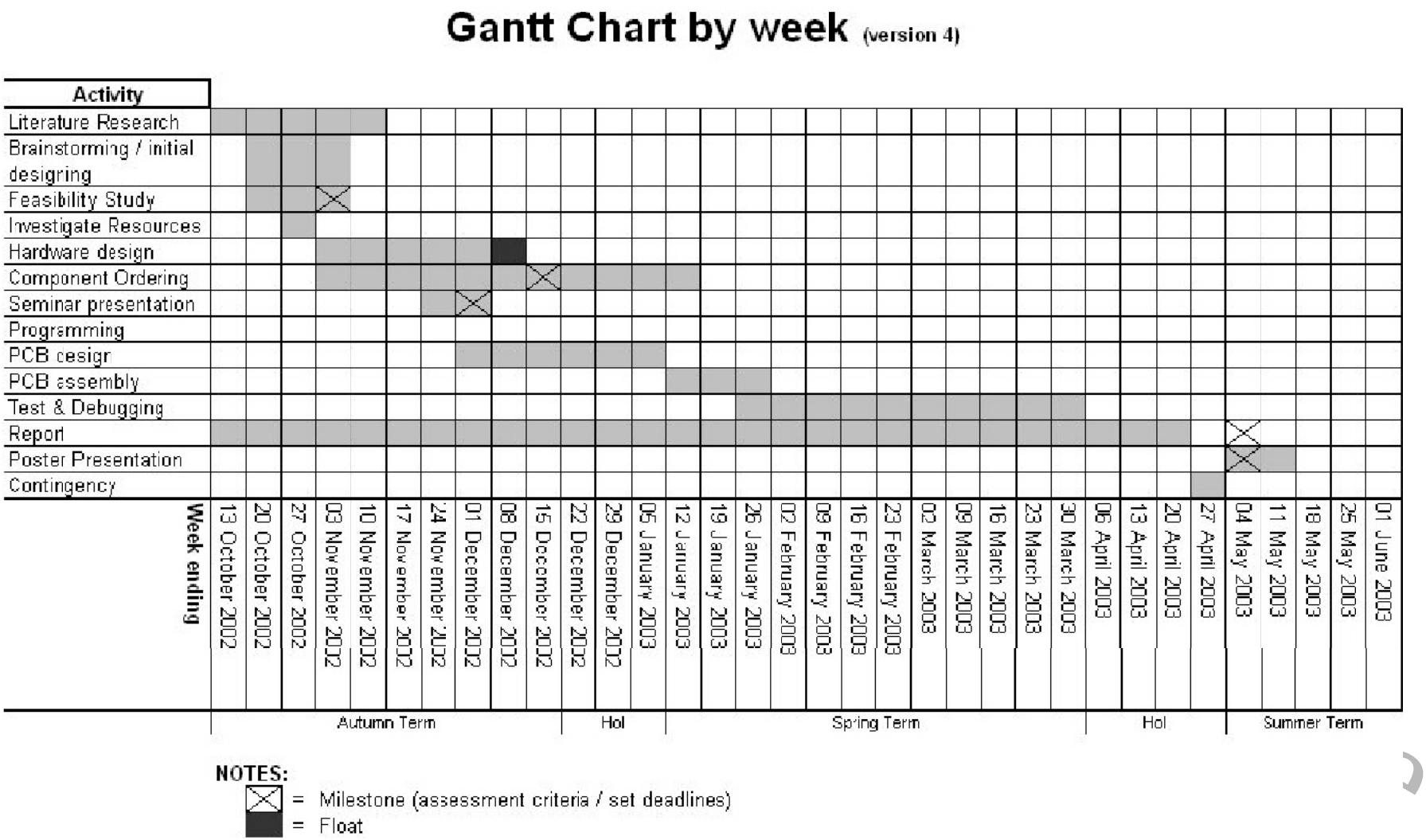
initial test plan had probably only allowed enough time to test a circuit board with very few or no faults. The result of this was that testing and debugging ran into the software development (programming) period and therefore no software could be developed during the project.

University of Hertfordshire

15.4 Initial Gantt chart



15.5 Final Gantt chart



16. References

1. MC68000 8-/16-/32-Bit Microprocessors User's Manual, MC68000UM, 9th Ed, Motorola Inc, p11-2
2. Coates, R, F: The Flight 68K MKII Training System User Manual, Flight Electronics Ltd, 3rd Ed, 1997, p4-1
3. Coates, R, F: The Flight 68K MKII Training System User Manual, Flight Electronics Ltd, 3rd Ed, 1997, p11-2
4. Flynn, Michael: Very High Speed Computing Systems, Proceedings of the IEE, Vol.54, 1966, pp 1901-1909

University of Hertfordshire

17. Bibliography

1. Catanzaro, Ben: Multiprocessor System Architectures, Sun Microsystems, 1994
2. Bacon, Jean: The Motorola MC68000, Prentice-Hall, 1986
3. Wilcox, Alan D: 68000 Microcomputer Systems, Prentice-Hall, 1987
4. Walter, Triebel A and Singh, Avtar: The 68000 and 68020 Microprocessors, Prentice-Hall, 1991
5. Kane, Gerry: 68000 Microprocessor Handbook, McGraw-Hill, 1981
6. Coates, R, F: The Flight 68K MKII Training System User Manual, Flight Electronics Ltd, 3rd Ed, 1997
7. Bolton, W: Microprocessor Systems, Longman, 2000
8. Clements, Alan: Microprocessor Systems; 68000 Hardware, Software and Interfacing, PWS, 1987
9. Culler, David E. Singh, Jaswinder Pal and Gupta, Anoop: Parallel Computer Architecture; A hardware/software approach, Morgan Kaufmann, 1999
10. Davies, Paul: The Indispensable Guide to C; With engineering applications, Addison-Wesley, 1995
11. Mimar, Tibet: Programming and Designing With The 68000 Family, Prentice-Hall, 1991
12. Tanenbaum, Andrew S: Operating Systems; Design and implementation, Prentice-Hall, 1987
13. Hwang, Kai. Briggs, Fayé A: Computer Architecture and Parallel Processing, McGraw-Hill, 1989
14. Stallings, William: Operating Systems, 2nd Ed, Prentice-Hall,
15. Duncan, Ralph: A Survey of Parallel Computer Architectures, IEEE Computer, February 1990, pp 5-16
16. Flynn, Michael: Some Computer Organisations and Their Effectiveness, IEEE Transactions on Computers, Vol.C-21, 1972, pp 94

University of Hertfordshire

17. Li, K, F. Dimopoulos, N, J. Atwood, J, W: The HM-Nucleus: distributed kernel design for the homogeneous multiprocessor, IEEE micro, 7(1), Feb 1987, pp 14-24
18. Boukerrou, R: Arbitration unit for microprocessor systems using a shared bus, Microprocessors and Microsystems, 12(4), May 1988, pp 211-213
19. MC68000 16-/32-Bit Microprocessor, MC68000/D, Motorola Inc, 1985
20. MC68000 8-/16-/32-Bit Microprocessors User's Manual, MC68000UM, 9th Ed, Motorola Inc.
21. M68000 Programmers Reference Manual, M68000PM, Motorola Inc, rev 1
22. Switch Fabric Implementation Using Shared Memory, AN1704, Motorola Inc, rev 1
23. Dual Port Memory for Multiprocessor Applications, AN1707, Motorola Inc, rev 0
24. ATM Switch with Shared Memory – A Simple Model, AN1299, Motorola Inc, rev 0
25. Technical Report Writing, The Institute of Electrical Engineers, 1997
26. Burns, R W, et al: Units and Symbols for Electrical and Electronic Engineering, The Institute of Electrical Engineers, 1992

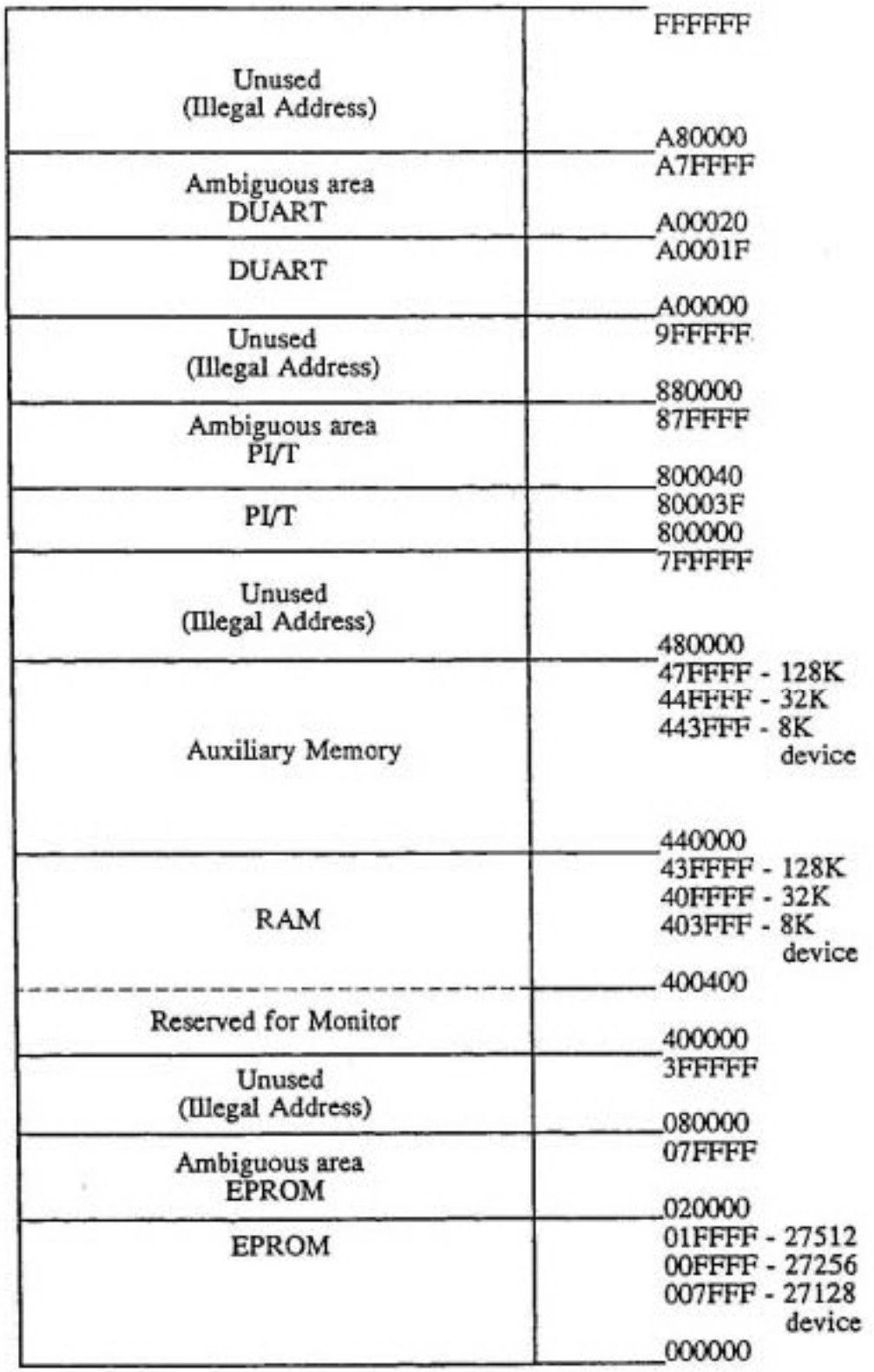
University of Hertfordshire

18. List of Appendices

Appendix A	Flight 68000 Memory Map	50
Appendix B	Flight 68000 Memory Map with Initial Shared RAM	51
Appendix C	Flight 68000 Memory Map with Final Shared RAM	52
Appendix D	Shared Memory Module Initial Circuit Diagram	53
Appendix E	Shared Memory Module Final Circuit Diagram	54
Appendix F	PCB Layout and Measurements	55
Appendix G	Parts List	56
Appendix H	Ram Access Time Calculations	58
Appendix I	Power Calculations for Initial Circuit	59
Appendix J	Power Calculations for Final Circuit	60
Appendix K	Address Bus Walking 1s Test	61
Appendix L	Data Bus Walking 1s Test	63
Appendix M	Flight 68000 Circuit Diagram	66
Appendix N	68000 Read/Write Cycle Timing Diagram	71
Appendix O	Users' Guide	72

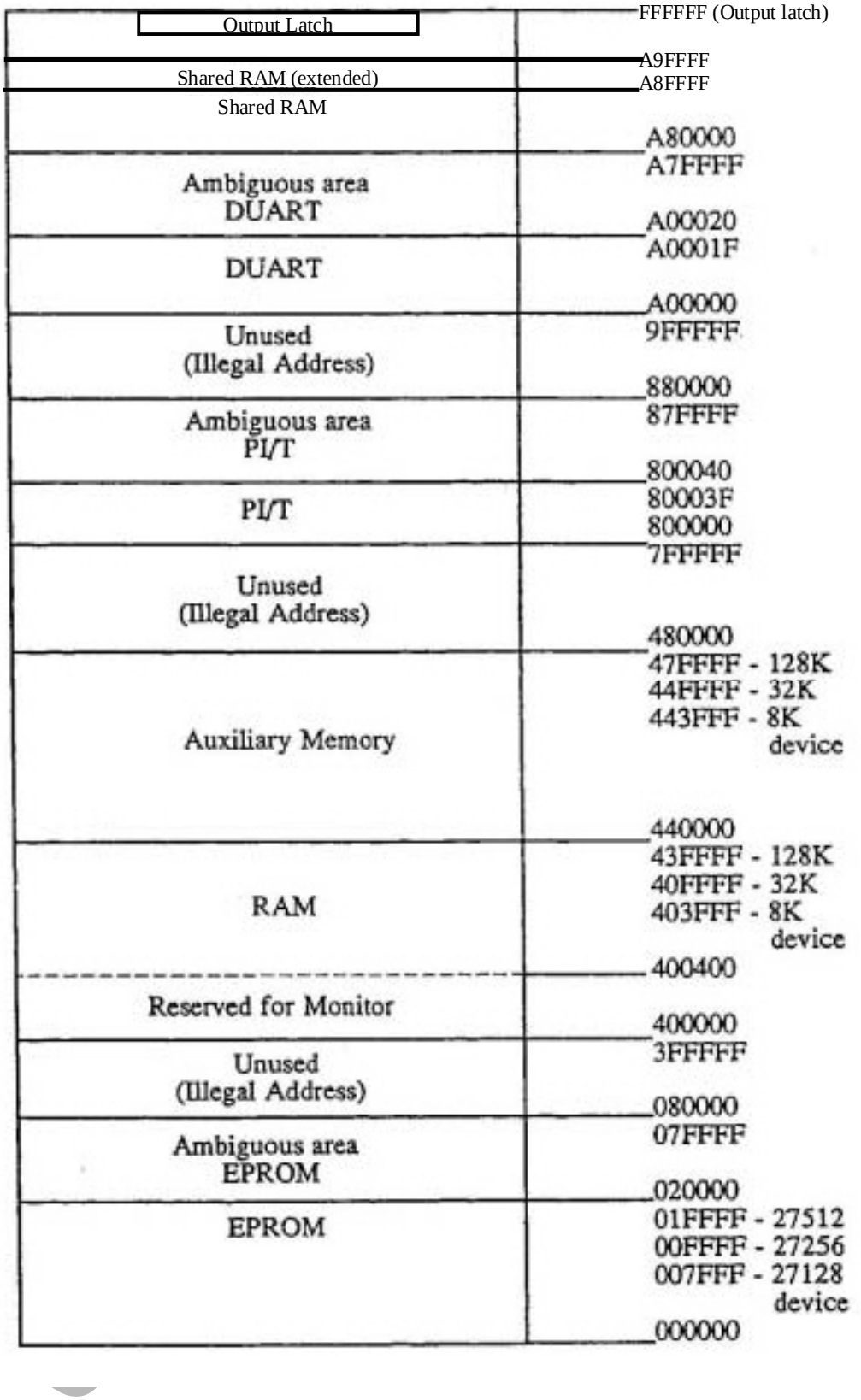
University of Hertfordshire

Appendix A. Flight 68000 Memory Map

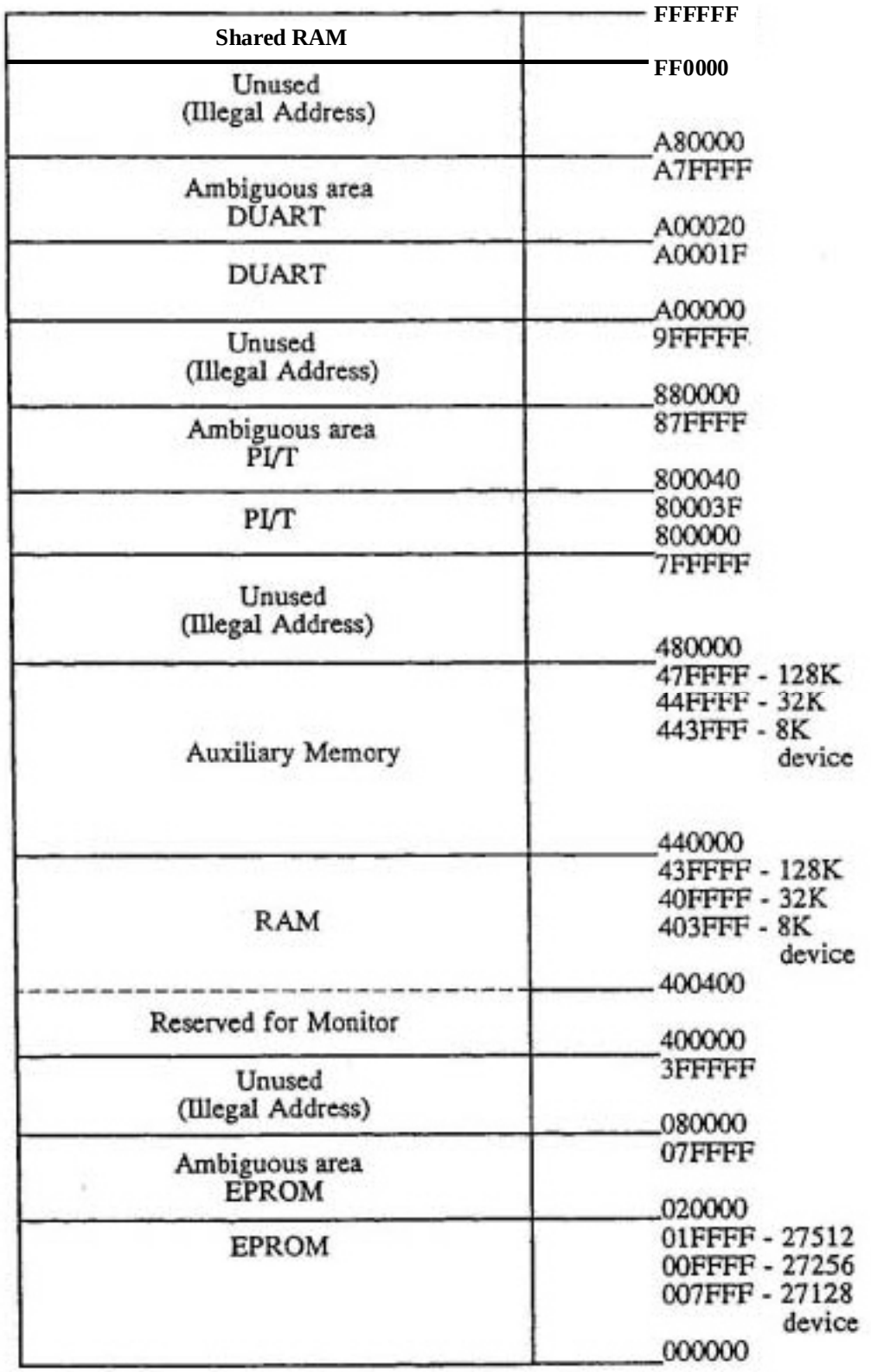


Coates, R, F: The Flight 68K MKII Training System User Manual, Flight Electronics Ltd, 3rd Ed, 1997, Appendix B

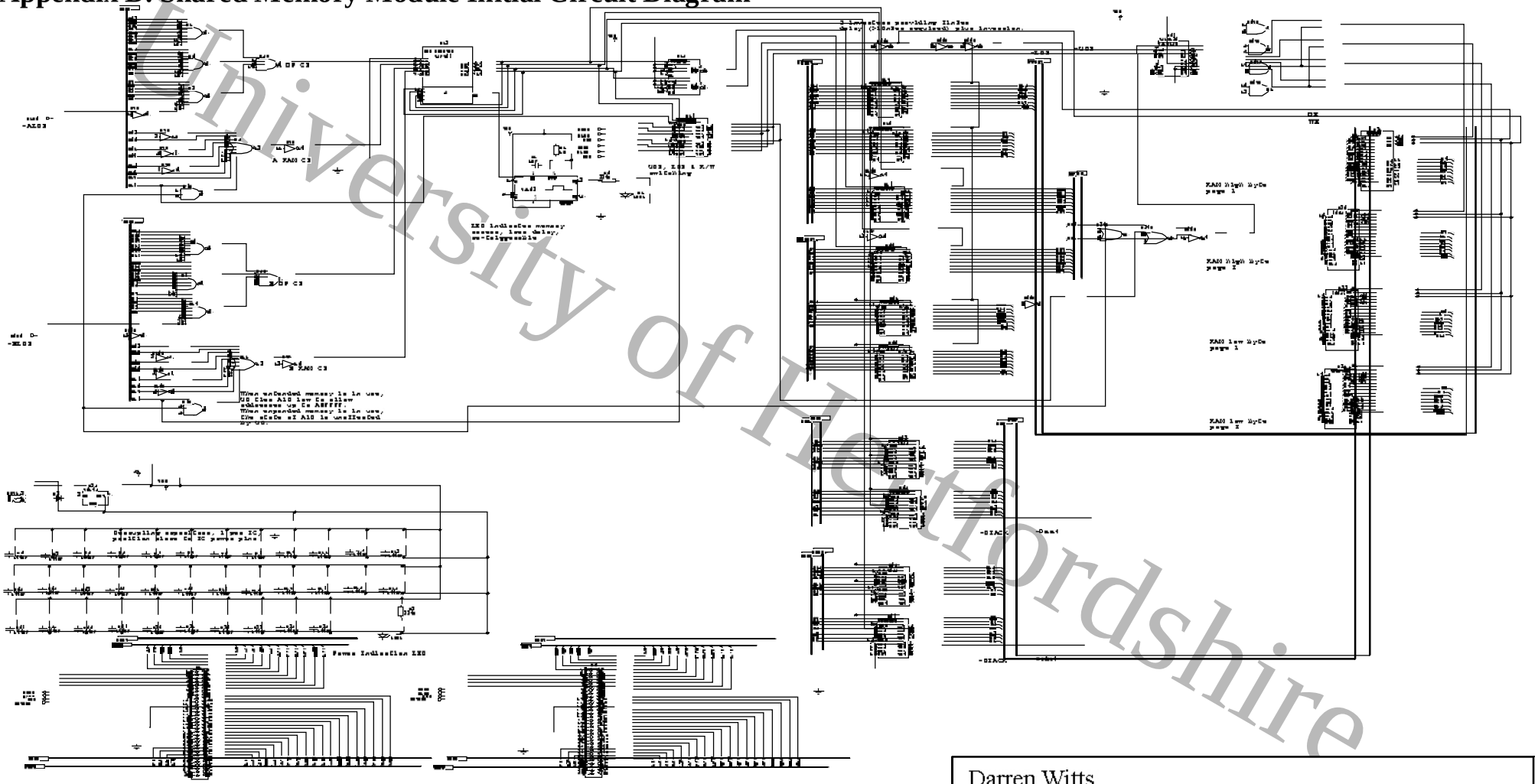
Appendix B. Flight 68000 Memory Map With Initial Shared RAM



Appendix C. Flight 68000 Memory Map With Final Shared RAM



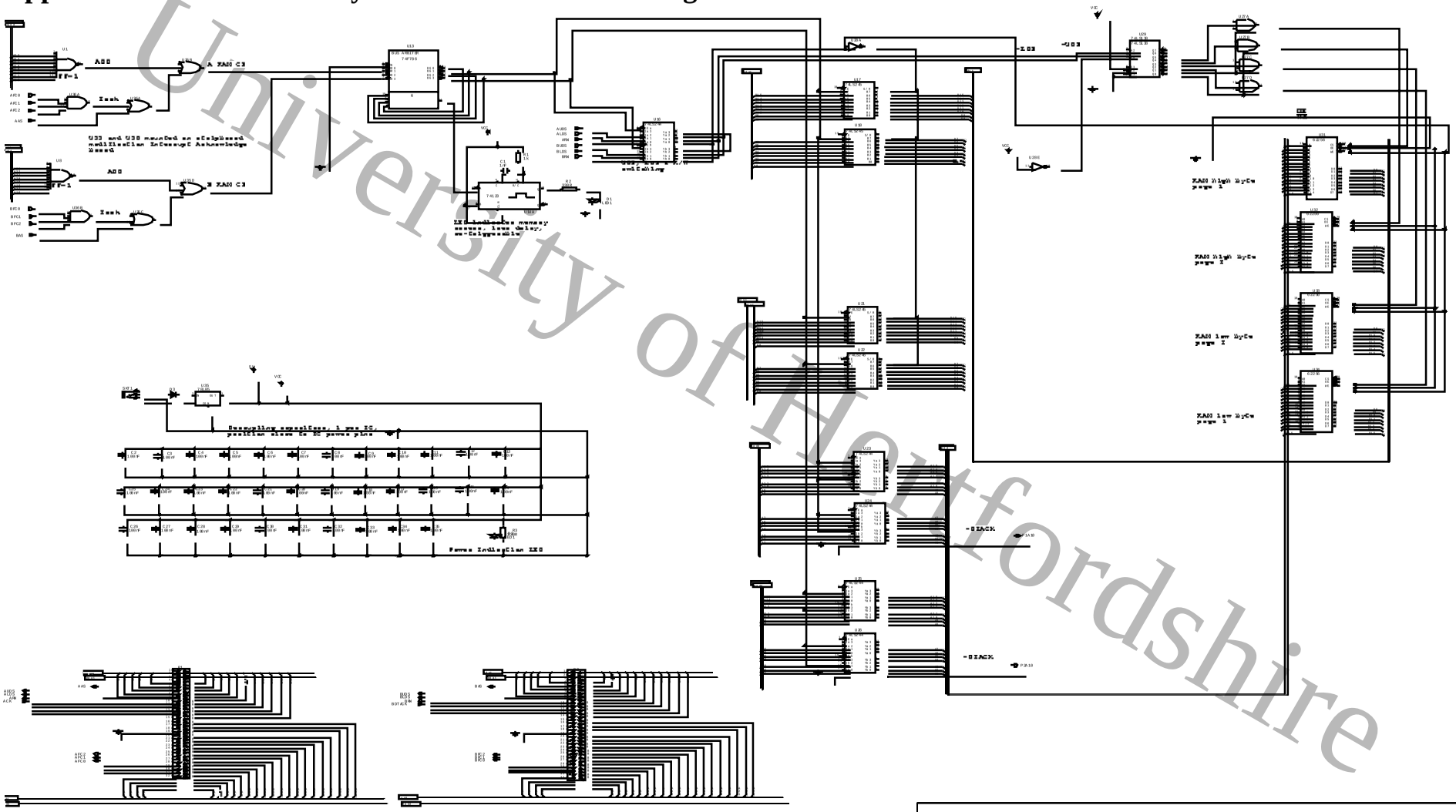
Appendix D. Shared Memory Module Initial Circuit Diagram



Darren Witts.
Flight 68000 Shared Memory Module Circuit Diagram
ver1, Nov 2002

University of Hertfordshire

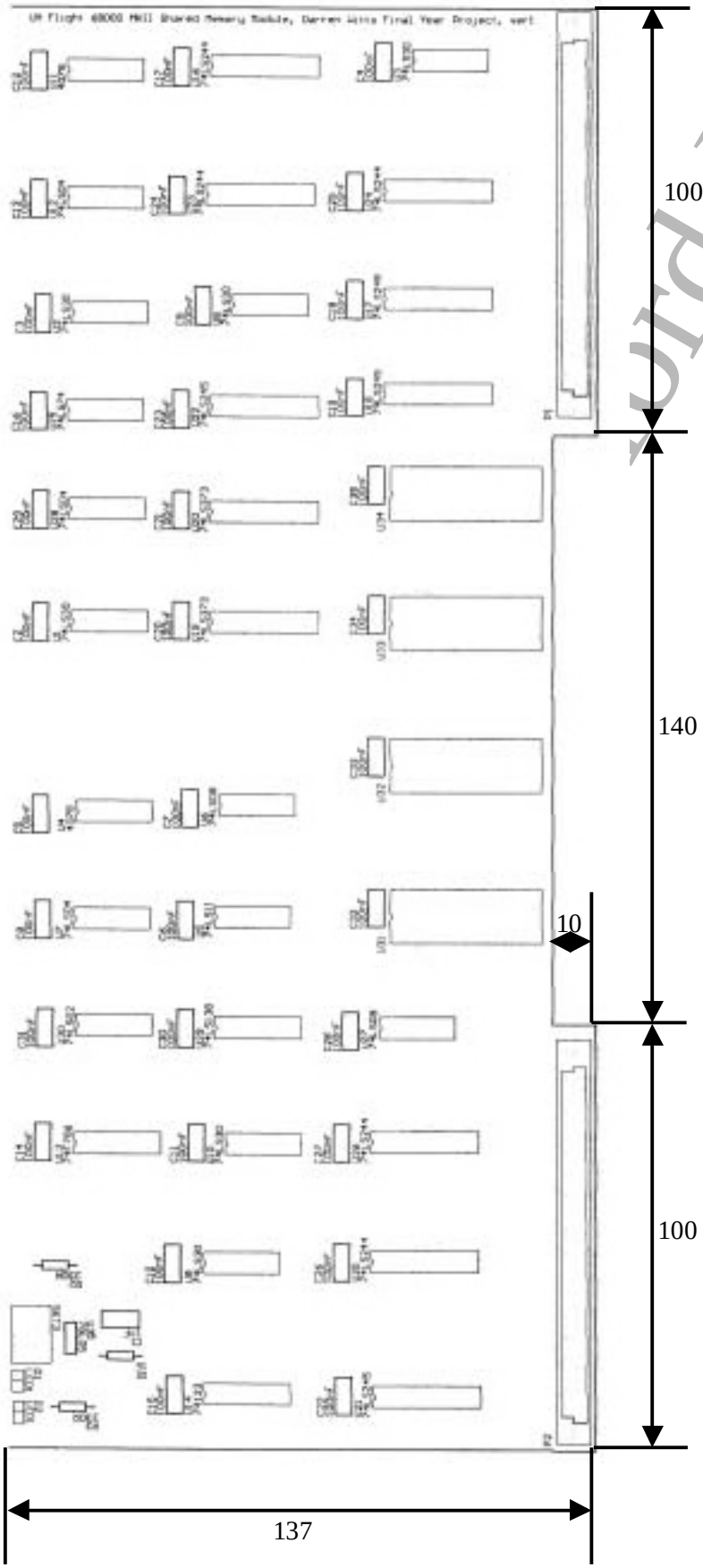
Appendix E. Shared Memory Module Final Circuit Diagram



University of Hertfordshire

Darren Witts.
Flight 68000 Shared Memory Module Circuit Diagram
ver2, Mar 2003

Appendix F. PCB Layout and Measurements



Dimensions: mm
Not to scale

Appendix G. Parts list (of parts used in initial and final circuit)

Item	Label/Value	Attributes	Designation	Supplier	Cost	31	100nF	RAD0.2	C31	UH	£0.02
1	1nF	RAD0.2	C1	UH	£0.02	32	100nF	RAD0.2	C32	UH	£0.02
2	100nF	RAD0.2	C2	UH	£0.02	33	100nF	RAD0.2	C33	UH	£0.02
3	100nF	RAD0.2	C3	UH	£0.02	34	100nF	RAD0.2	C34	UH	£0.02
4	100nF	RAD0.2	C4	UH	£0.02	35	100nF	RAD0.2	C35	UH	£0.02
5	100nF	RAD0.2	C5	UH	£0.02	36	LED1	SIP2	D1	UH	£0.04
6	100nF	RAD0.2	C6	UH	£0.02	37	LED2	SIP2	D2	UH	£0.03
7	100nF	RAD0.2	C7	UH	£0.02	38	1N4004	DIODE0.4	D3	UH	£0.01
8	100nF	RAD0.2	C8	UH	£0.02	39	1k	AXIAL0.4	R1	UH	£0.02
9	100nF	RAD0.2	C9	UH	£0.02	40	330R	AXIAL0.4	R2	UH	£0.02
10	100nF	RAD0.2	C10	UH	£0.02	41	330R	AXIAL0.4	R3	UH	£0.02
11	100nF	RAD0.2	C11	UH	£0.02	42	2.1 DC SKT		SKT1	MAPLIN	£0.49
12	100nF	RAD0.2	C12	UH	£0.02	43	74HC30	DIP14	U1	UH	£0.14
13	100nF	RAD0.2	C13	UH	£0.02	44	74HC30	DIP14	U2	UH	£0.14
14	100nF	RAD0.2	C14	UH	£0.02	45	74HC30	DIP14	U3	UH	£0.14
15	100nF	RAD0.2	C15	UH	£0.02	46	74HC4078	DIP14	U4	UH	£0.16
16	100nF	RAD0.2	C16	UH	£0.02	47	74HC11	DIP14	U5	UH	£0.14
17	100nF	RAD0.2	C17	UH	£0.02	48	74HC08	DIP14	U6	UH	£0.14
18	100nF	RAD0.2	C18	UH	£0.02	49	74HC04	DIP14	U7	UH	£0.14
19	100nF	RAD0.2	C19	UH	£0.02	50	74HC30	DIP14	U8	UH	£0.14
20	100nF	RAD0.2	C20	UH	£0.02	51	74HC30	DIP14	U9	UH	£0.14
21	100nF	RAD0.2	C21	UH	£0.02	52	74HC30	DIP14	U10	UH	£0.14
22	100nF	RAD0.2	C22	UH	£0.02	53	73HC4078	DIP14	U11	UH	£0.16
23	100nF	RAD0.2	C23	UH	£0.02	54	74HC04	DIP14	U12	UH	£0.14
24	100nF	RAD0.2	C24	UH	£0.02	55	74F786	DIP16	U13	EXTERNAL	£6.00
25	100nF	RAD0.2	C25	UH	£0.02	56	74HC123	DIP16	U14	UH	£0.23
26	100nF	RAD0.2	C26	UH	£0.02	57	74HC74	DIP14	U15	UH	£0.17
27	100nF	RAD0.2	C27	UH	£0.02	58	74HC244	DIP20	U16	UH	£0.30
28	100nF	RAD0.2	C28	UH	£0.02	59	74HC245	DIP20	U17	UH	£0.30
29	100nF	RAD0.2	C29	UH	£0.02	60	74HC245	DIP20	U18	UH	£0.30
30	100nF	RAD0.2	C30	UH	£0.02	61	74HC373	DIP20	U19	UH	£0.30

University of Hertfordshire

62	74HC373	DIP20	U20	UH	£0.30
63	74HC245	DIP20	U21	UH	£0.30
64	74HC245	DIP20	U22	UH	£0.30
65	74HC244	DIP20	U23	UH	£0.30
66	74HC244	DIP20	U24	UH	£0.30
67	74HC244	DIP20	U25	UH	£0.30
68	74HC244	DIP20	U26	UH	£0.30
69	74HC08	DIP14	U27	UH	£0.14
70	74HC04	DIP14	U28	UH	£0.14
71	74HC138	DIP16	U29	UH	£0.23
72	74HC02	DIP14	U30	UH	£0.14
73	62256	DIP28	U31	UH	£3.30
74	62256	DIP28	U32	UH	£3.30
75	62256	DIP28	U33	UH	£3.30
76	62256	DIP28	U34	UH	£3.30
77	78L05	TO-126	U35	UH	£0.14
78	74HC08	DIP14	U36	UH	£0.14
79	74HC32	DIP14	U37	UH	£0.14
80	DIN41612	TYPE Q	P1	RS	£4.73
81	DIN41612	TYPE Q	P2	RS	£4.73
82	PCB			PCBPool	£123.23
83	DIL SOCKETS	DIP14		UH	£0.48
84	DIL SOCKETS	DIP16		UH	£0.09
85	DIL SOCKETS	DIP20		UH	£0.44
86	DIL SOCKETS	DIP28		UH	£0.24
				TOTAL:	**£160.96**

University of Hertfordshire

Appendix H. RAM access time calculations

Time to access shared memory = $t_{U7} + t_{U4} + t_{U7} + t_{U13} + t_{buffers} + t_{CS} + t_{RAM}$

Where $t_{CS} = MAX(t_{U28}+t_{U28}+t_{U29}\ ,\ t_{U29}+t_{U27})$ and

$t_{buffers} = MAX(t_{U16}\ ,t_{U17}\ ,t_{U18}\ ,t_{U23}\ ,t_{U24})$, due to parallel paths.

The following table shows the device propagation delays.

Device Designation	Device Type	Time Designation	Time (*nSec*)
U7	74HC04	t_{U7}	7
U4	74HC4078	t_{U4}	26
U13	74F786	t_{U13}	6.6
U16,23,24,25,26	74HC244	t_{U16}	9
U17,18,21,22	74HC245	t_{U17}	7
U27	74HC08	t_{U27}	7
U19,20	74HC373	t_{U19}	12
U28	74HC04	t_{U28}	7
U29	74HC138	t_{U29}	12
RAM	62256	t_{RAM}	See below

From the above table:

$t_{CS} = MAX(t_{U28}+t_{U28}+t_{U29}\ ,\ t_{U29}+t_{U27})=MAX(7+7+12\ ,\ 12+7)=MAX(26,19)=26nSec$

$t_{buffers} = MAX(t_{U16}\ ,t_{U17}\ ,t_{U18}\ ,t_{U23}\ ,t_{U24})=MAX(9, 7, 7, 9, 9)=9nSec$

Therefore, time to access RAM =

$$t_{U7} + t_{U4} + t_{U7} + t_{U13} + t_{buffers} + t_{CS} + t_{RAM} = 7 + 26 + 7 + 6.6 + 9 + 26 + t_{RAM} = 81.6 + t_{RAM}$$

The RAM used must therefore have a maximum access time determined by:

t_{RAM_MAX} = 400 - 81.6 = **318.4*nSec***

Where 400*nSec* = read/write bus cycle of the Flight 68000 (10*MHz* clock).

University of Hertfordshire

Appendix I. Power Calculations for Initial Circuit

Current Drawn

Designation	Device	Number of	Max Current *mA*	Total (*mAmps*)
U1,2,3,8,9,10	74HC30	6.00	50.000	300.000
U4,11	74HC4078	2.00	0.001	0.002
U5	74HC11	1.00	50.000	50.000
U6,27	74HC08	2.00	50.000	100.000
U7,12,28	74HC04	3.00	50.000	150.000
U13	74F786	1.00	55.000	55.000
U14	74HC123	1.00	50.000	50.000
U15	74HC74	1.00	50.000	50.000
U16,23,24,25,26	74HC244	5.00	50.000	250.000
U17,18,21,22	74HC245	4.00	70.000	280.000
U19,20	74HC373	2.00	70.000	140.000
U29	74HC138	1.00	50.000	50.000
U30	74HC02	1.00	50.000	50.000
U31,32,33,34	62256	4.00	15.000	60.000
			TOTAL CURRENT:	**1585.002**

Voltage

+7.5*Volts* to 12*Volts* DC

Power

At 5Volts = 5*1.6 = 8*Watts*
At 7.5Volts = 7.5*1.6 = 12*Watts*
At 12Volts = 12*1.6 = 19.2*Watts*

Measured Current

The current drawn by the circuit was measured with the circuit under low load conditions (memory access initiated and DTACK disconnected in order to continuously enable buffers and stop devices entering a quiescent state).

Measured Current = 0.5*Amps*

The maximum current specified in the table above does not take into account that some of the circuitry is idle when others are not (i.e. Port 1 buffers not enabled when Port 2 buffers are due to arbitration) and the quiescent current of the disabled devices would be lower and will account for some of the reduction in calculated power from measured power.

University of Hertfordshire

Appendix J. Power Calculations for Final Circuit

Current Drawn

Designation	Device	Number of	Max Current mA	Total (mAmps)
U1,2,3,8,9,10	74HC30	0.00	50.000	0.000
U4,11	74HC4078	2.00	0.001	0.002
U5	74HC11	0.00	50.000	0.000
U6,27	74HC08	2.00	50.000	100.000
U7,12,28	74HC04	3.00	50.000	150.000
U13	74F786	1.00	55.000	55.000
U14	74HC123	1.00	50.000	50.000
U15	74HC74	1.00	50.000	50.000
U16,23,24,25,26	74HC244	5.00	50.000	250.000
U17,18,21,22	74HC245	4.00	70.000	280.000
U19,20	74HC373	0.00	70.000	0.000
U29	74HC138	1.00	50.000	50.000
U30	74HC02	0.00	50.000	0.000
U31,32,33,34	62256	4.00	15.000	60.000
			TOTAL CURRENT:	**1045.002**

Voltage

+7.5*Volts* to 12*Volts* DC

Power

At 5Volts = 5*1.05 = 5.25*Watts*
At 7.5Volts = 7.5*1.05 = 7.9*Watts*
At 12Volts = 12*1.05 = 12.6*Watts*

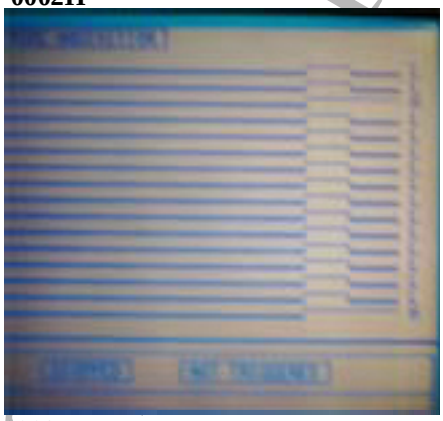
University of Hertfordshire

Appendix K. Address Bus Walking 1s Test Waveforms

0000H

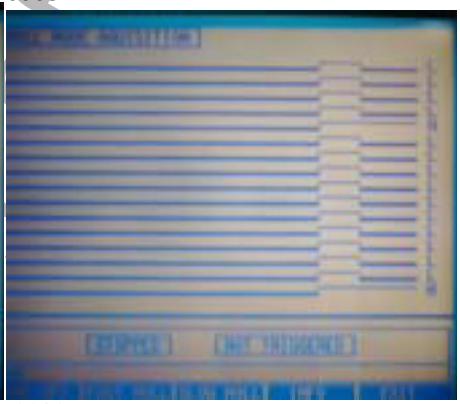
0002H

0004H



0008H

0010H



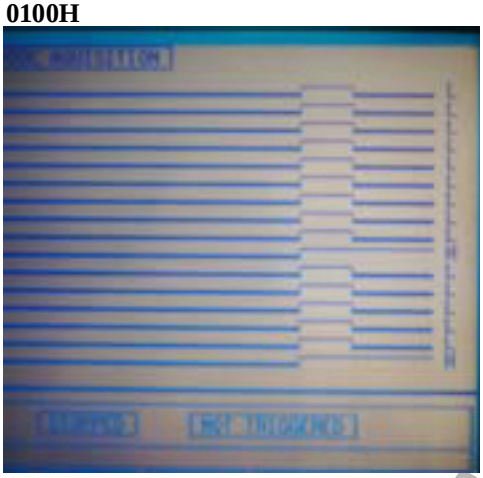
0020H

0040H

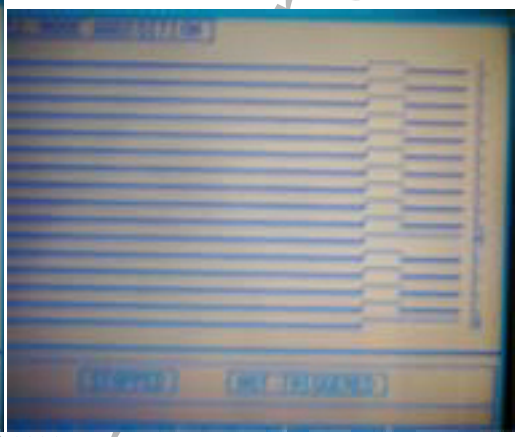
0080H

0100H

0200H

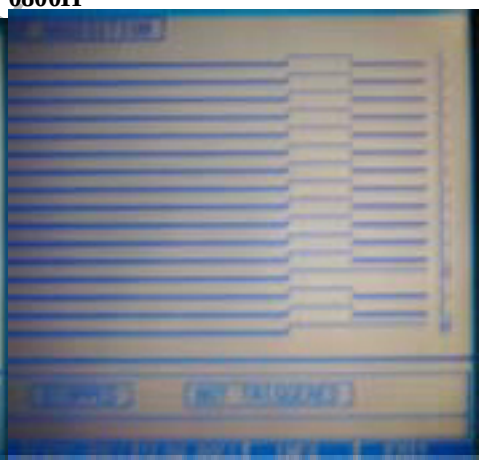


0400H

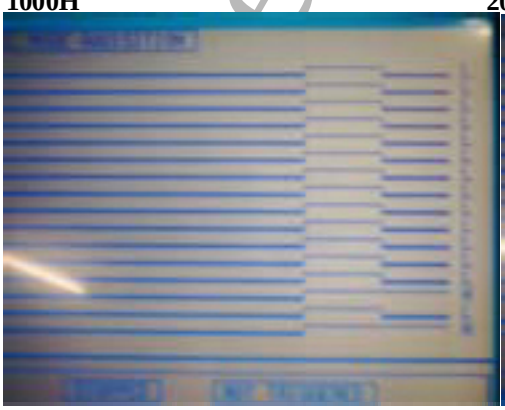


0800H

1000H



2000H



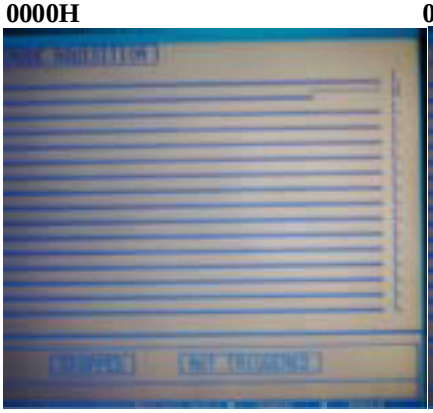
4000H

8000H

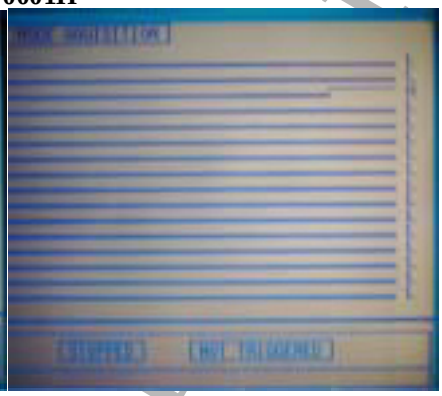
Appendix L. Data Bus Walking 1s Test Waveforms

0000H

0001H



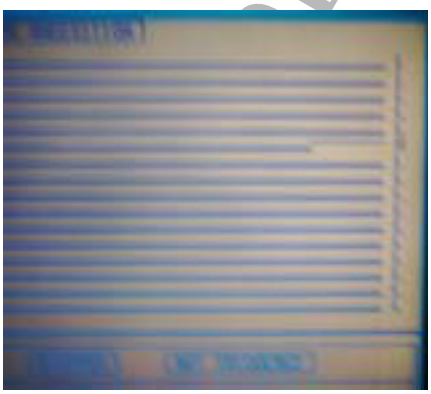
0002H



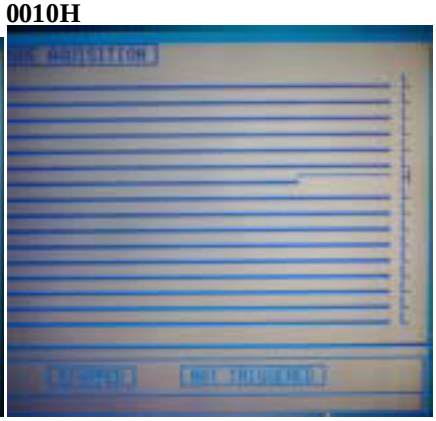
0004H

0008H

0010H



0020H



0040H

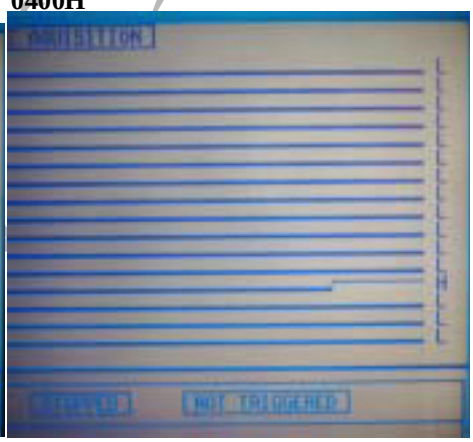
0080H

0100H

0200H

0400H

0800H



1000H

2000H

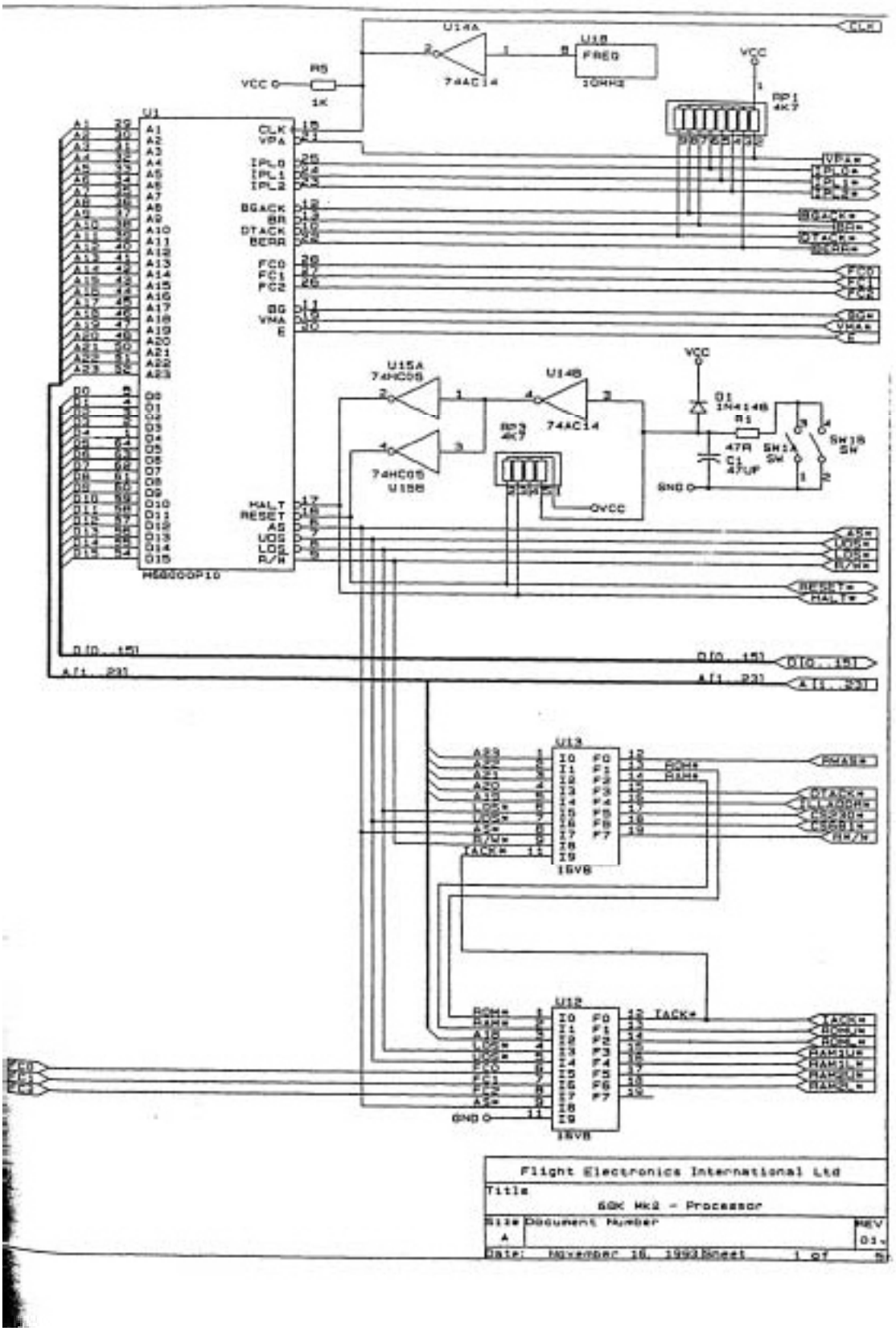
4000H

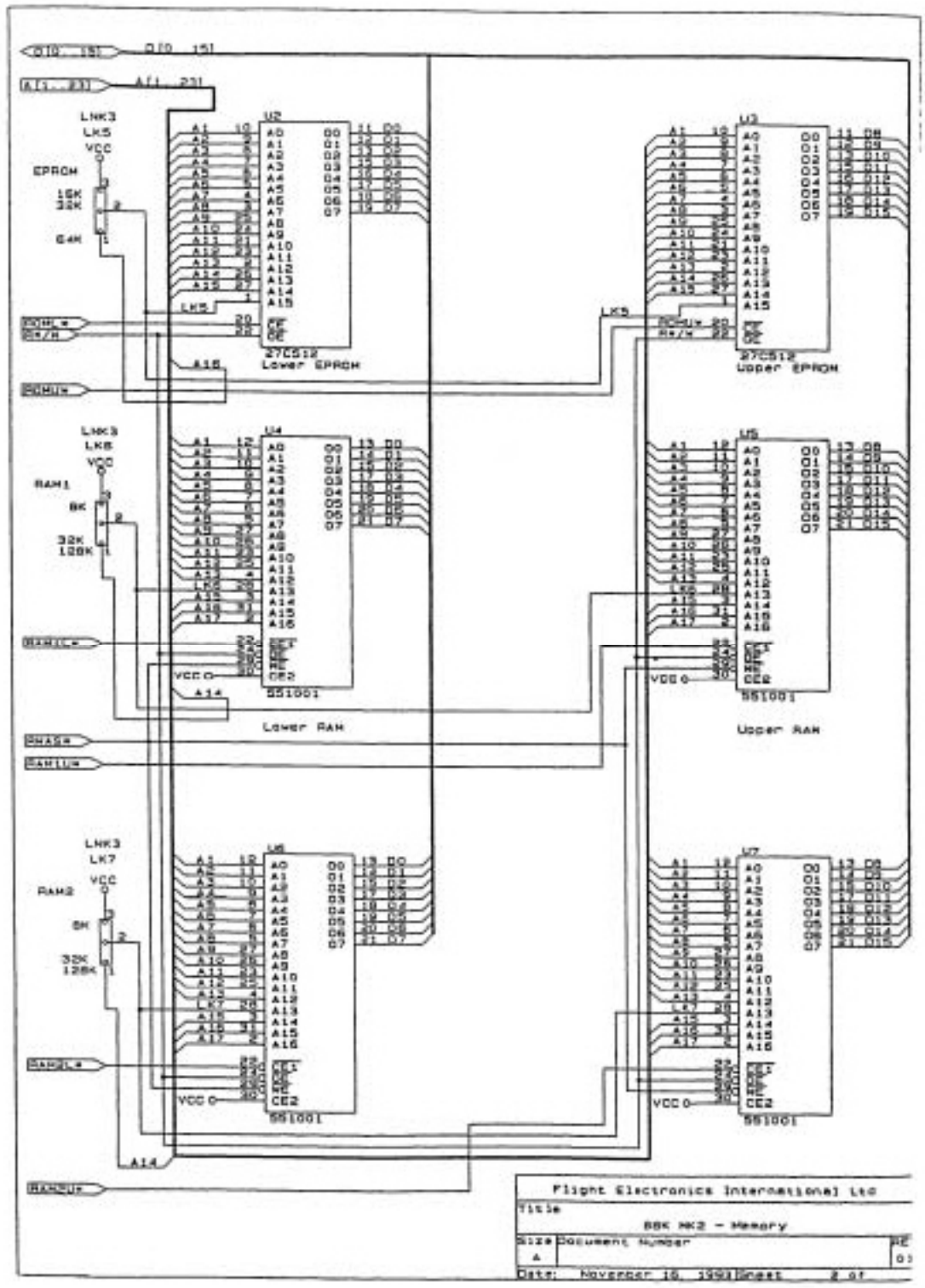
8000H

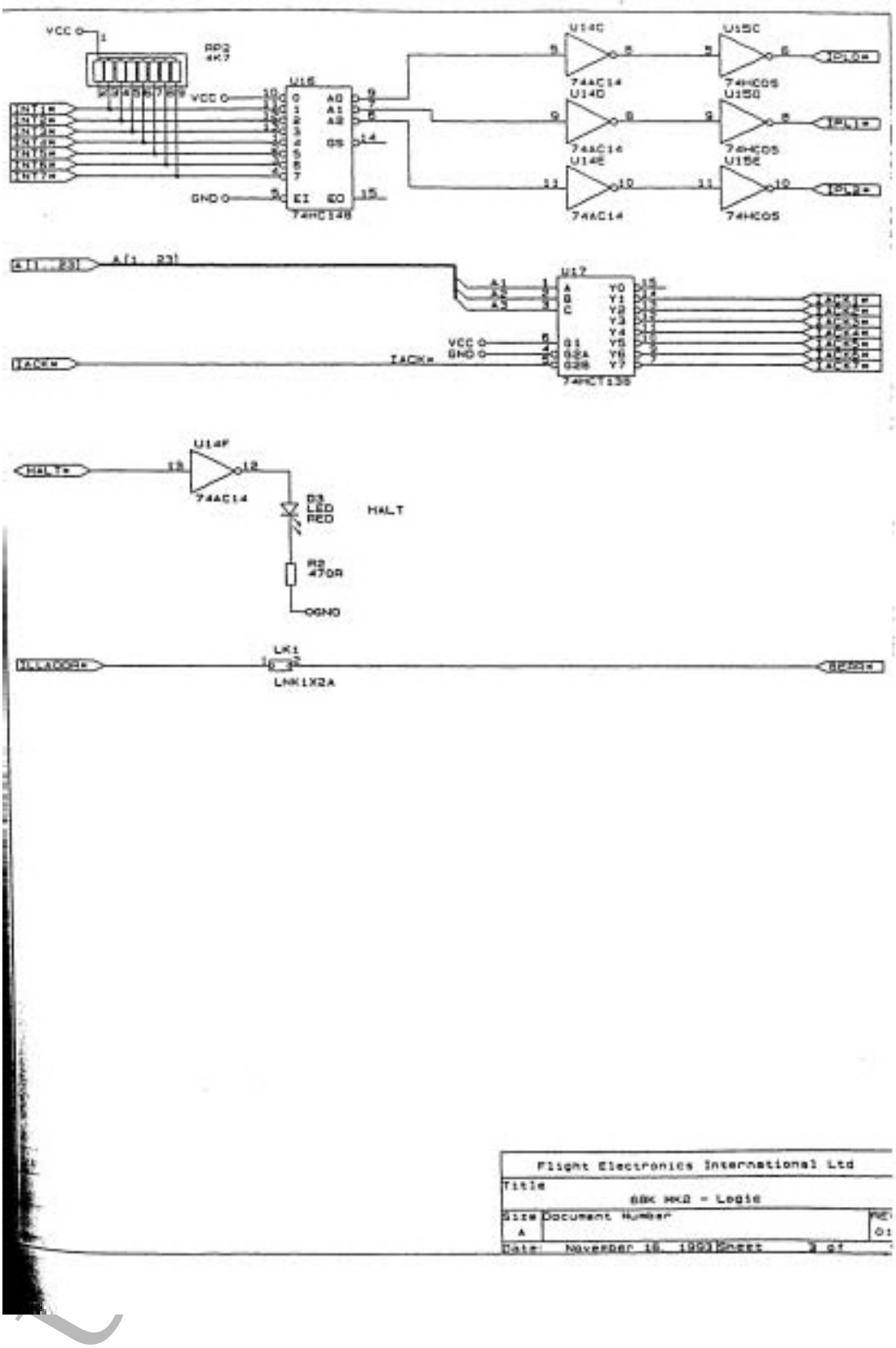
University of Hertfordshire

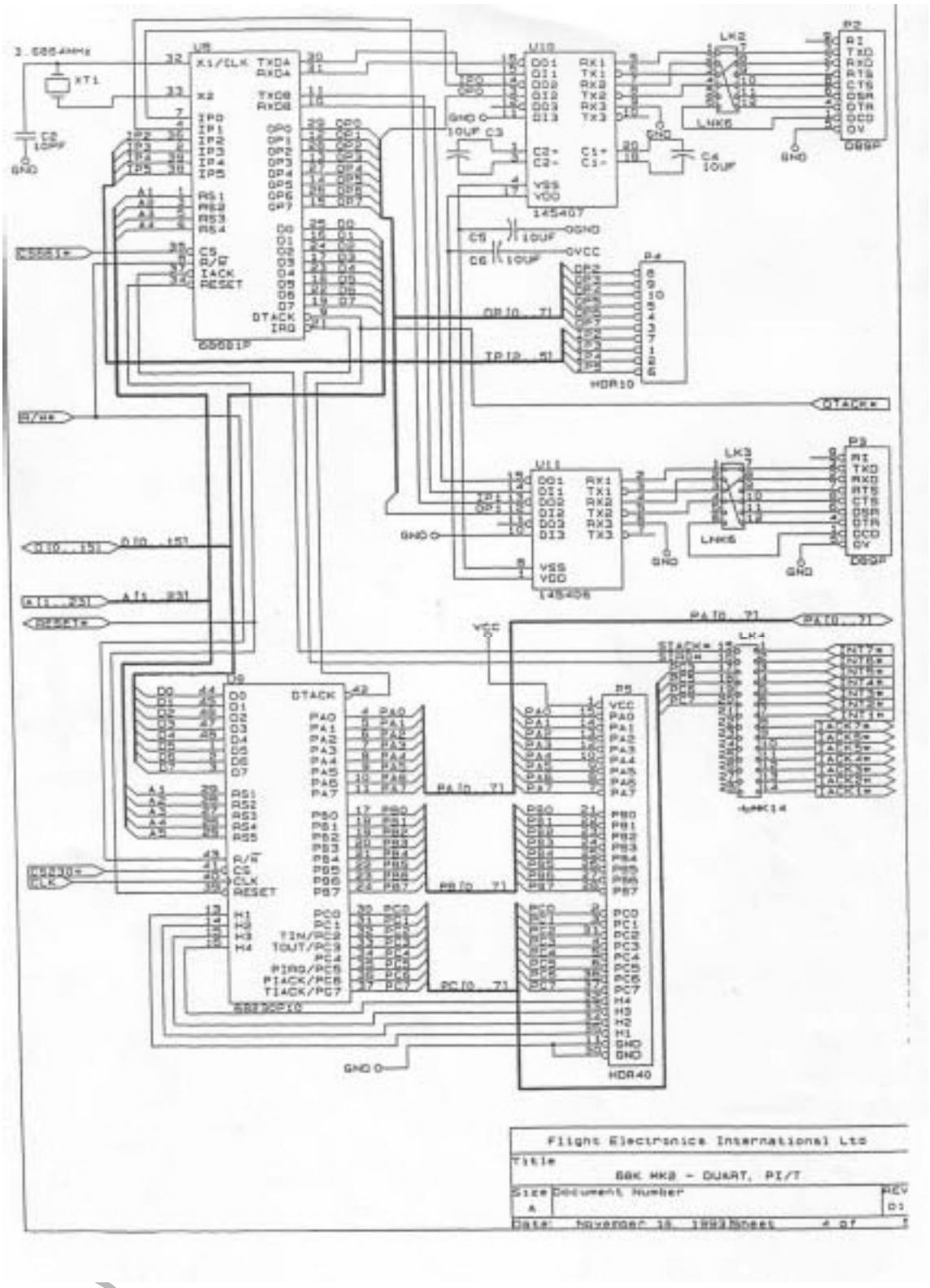
Appendix M. Flight 68000 Circuit Diagram

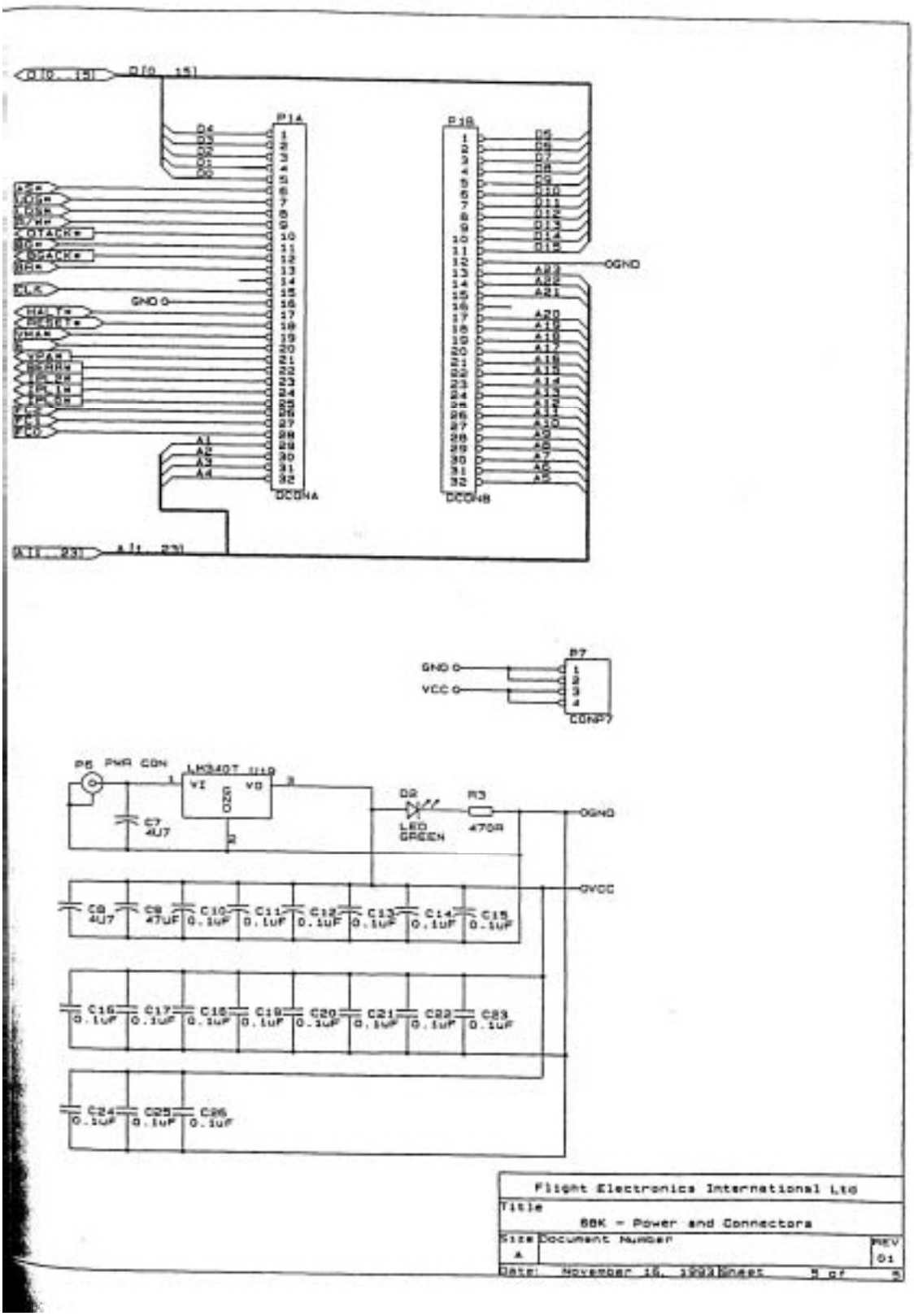
Coates, R, F: The Flight 68K MKII Training System User Manual, Flight Electronics Ltd, 3rd Ed, 1997, Appendix F



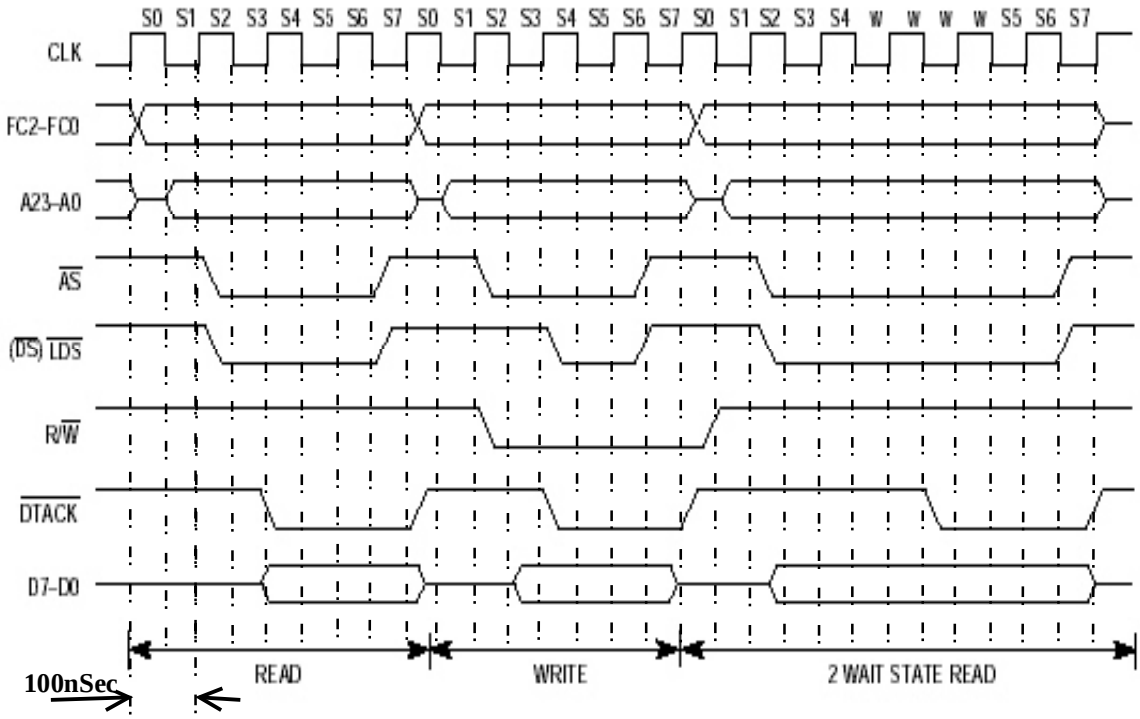

LNK3
LK5
VCC
EPROM
16K
32K
64K
LK5
27C512
Lower EPROM
27C512
Upper EPROM
LNK3
LK6
VCC
RAM1
8K
32K
128K
551001
Lower RAM
551001
Upper RAM
LNK3
LK7
VCC
RAM2
8K
32K
128K
551001
551001
A14
Flight Electronics International Ltd
Title
88K MK2 - Memory
Size
A
Document Number
Date: November 10, 1993
Sheet 2 of


VCC
RP2
4K7
U16
74HC148
GND
U14C
74AC14
U15C
74HCO5
U14D
74AC14
U15D
74HCO5
U14E
74AC14
U15E
74HCO5
U17
74HCT138
U14F
74AC14
D2
LED
RED
HALT
R2
470R
LK1
LNK1X2A
Flight Electronics International Ltd
Title
68K MK2 - Logic
Size
A
Document Number
Date: November 18, 1993
Sheet


Flight Electronics International Ltd
Title
68K MK3 - DUART, PI/T
Size
A
Document Number
REV
D1
Date: November 18, 1993 Sheet 4 of 7


P14
P18
DCON4A
DCON4B
OGND
GND
A[1..23]
P7
CONP7
P6 PWR CON
LM340T-5.0
C7 4U7
D2 LED GREEN
R3 470R
OVCC
C8 4U7
C9 47uF
C10 0.1uF
C11 0.1uF
C12 0.1uF
C13 0.1uF
C14 0.1uF
C15 0.1uF
C16 0.1uF
C17 0.1uF
C18 0.1uF
C19 0.1uF
C20 0.1uF
C21 0.1uF
C22 0.1uF
C23 0.1uF
C24 0.1uF
C25 0.1uF
C26 0.1uF
Flight Electronics International Ltd
Title
68K - Power and Connectors
Size A
Document Number
REV 01
Date: November 18, 1993 Sheet 5 of 5

Appendix N. 68000 Read / Write Cycle Timing Diagram



From Motorola 8-/16-/32-Bit Microprocessors User's Manual, 9th Ed, 1993, p4-2

University of

Appendix O. Users' Guide

Those wishing to use the shared memory module should follow the following simple instructions.

1. Connect the shared memory module to two Flight 68000 MK-II microprocessor training systems via connectors P1 and P2. These connectors are keyed so that incorrect orientation is not possible. Ensure the connectors are pushed down onto the microprocessor board firmly.
2. Remove link LK1 from the Flight 68000 board
3. Insert the Flight monitor EPROMs if not already fitted
4. Application software can address the 64*kbytes* of shared memory, between FF0000 and FFFFFF

University of Hertfordshire